# 镜像的导入导出

## 镜像导入导出

### docker save[1]

#### 命令描述

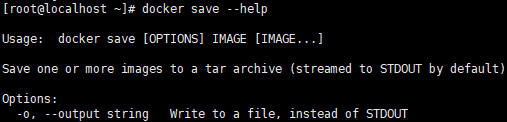


图 1‑1 docker save 命令帮助信息

docker save 命令是将**一个**或**多个** Docker 镜像（images）保存到 tar 文件中(默认情况下流到 STDOUT 标准输出)。生成给的 tar 文件会包含镜像的所有父级镜像层（parent layers）以及所有 tab version。[1]docker save 命令创建的镜像备份可以通过 docker load 命令重新加载到 Docker 中。

#### 使用格式

docker save [OPTIONS] IMAGE [IMAGE...]

#### 命令选项

表 1‑1 docker save 命令选项

|  |  |  |
| --- | --- | --- |
| Name, shorthand | Default | Description |
| --output, -o [FilePath] |  | 镜像导出从 STDOUT（标准输出）重定向到指定文件中 |

#### 使用例子

##### 创建可以直接被 docker load 命令操作的 tar 文件

$ docker save -o nginx.tar nginx:latest

$ ls -sh nginx.tar

125M nginx.tar

$ docker save -output redis.tar redis

$ ls -sh redis.tar

97M redis.tar

$ docker save > mysql.tar mysql:5.7

$ ls -sh mysql.tar

422M mysql.tar

##### 镜像导出成 gzip 压缩后的 tar.gz 压缩文件

我们可以使用 gzip 压缩让导出后的文件更小，不过导出消耗时间也会更长。

$ docker save mysql:5.7 | gzip > mysql.tar.gz

$ ls -sh mysql.tar.gz

141M mysql.tar.gz

在我的 1 核虚拟机下使用 gzip 与否导出命令执行时间比较如下：

$ time docker save -o mysql.tar mysql:5.7

real 0m9.442s

user 0m0.164s

sys 0m0.545s

$ time docker save mysql:5.7 | gzip > mysql.tar.gz

real 1m20.389s

user 1m8.024s

sys 0m1.445s

##### 批量打包

docker save 命令是支持将多个镜像打包到一个 tar 文件中的：

$ docker save -o all.tar redis:latest nginx:latest mysql:5.7

$ docker load -i all.tar

Loaded image: redis:latest

Loaded image: nginx:latest

Loaded image: mysql:5.7

### docker load[2]

#### 命令描述

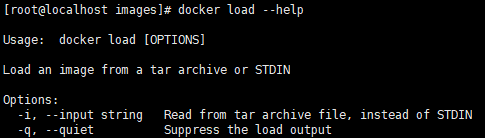


图 1‑2 docker load 命令帮助信息

docker load 命令是从一个 tar 文件或者是 STDIN（标准输入）中读取一个 Docker 镜像。docker load 命令可以直接从 gzip、bzip2 或者 xz 压缩的 tar 文件中导入镜像。[2]

#### 使用格式

docker load [OPTIONS]

#### 命令选项

表 1‑2 docker load 命令选项

|  |  |  |
| --- | --- | --- |
| Name, shorthand | Default | Description |
| --input, -i [FilePath] |  | 代替 STDIN（标准输入）从指定 tar 文件中读取一个镜像 |
| --quit, -q |  | 静默模式——抑制加载过程中的输出 |

#### 使用例子

$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

$ docker load -i redis.tar.gz

Loaded image: redis:latest

$ docker load < nginx.tar

Loaded image: nginx:latest

$ docker load --input mysql.tar

Loaded image: mysql:5.7

$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

nginx latest 540a289bab6c 5 weeks ago 126MB

mysql 5.7 cd3ed0dfff7e 6 weeks ago 437MB

redis latest de25a81a5a0b 6 weeks ago 98.2MB

## 容器导入导出

### docker export[3]

#### 命令描述

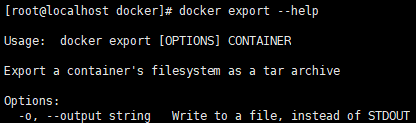


图 1‑3 docker export 帮助信息

docker export 是将**一个** Docker 容器（container）的文件系统导出为 tar 文件。docker export 命令并不会导出与容器关联的数据卷（volume）的内容。如果一个卷被挂载在容器中现有目录的顶部，docker export 将导出底层目录的内容，而不是卷的内容。[3]

#### 使用格式

docker export [OPTIONS] CONTAINER

#### 命令选项

表 1‑3 docker export 命令选项

|  |  |  |
| --- | --- | --- |
| Name, shorthand | Default | Description |
| --output, -o [FilePath] |  | 容器导出从 STDOUT（标准输出）重定向到指定文件中 |

#### 使用例子

下面命令有着相同的结果：

$ docker export red\_panda > latest.tar

$ docker export –output latest.tar red\_panda

$ docker export -o latest.tar red\_panda

### docker import[4]

#### 命令描述

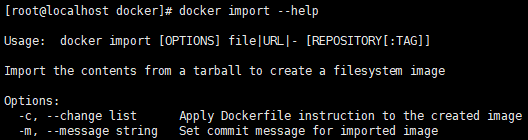


图 1‑4 docker import 命令帮助信息

docker import 命令用于从一个 tar 压缩文件中导入一个文件系统镜像（image）。我们可以指定一个 URL 或者 -（破折号）直接从 STDIN 获取数据。这个 URL 可以指向一个包含文件系统或 Docker 主机上的单个文件（.tar、.tar.gz、.tgz、.bzip、.tar.xz 或者 .txz）。如果您指定了一个归档文件，Docker 会在容器中相对于 /（root）解压缩它。如果指定单个文件，则必须指定主机内的完整路径。要从远程位置导入，请指定以 http:// 或 https:// 协议开头的 URI。[4]

--change 选项将对所创建的镜像执行 Dockerfile 指令。目前支持的 Dockerfile 执行如下：[4]

* CMD
* ENTRYPOINT
* ENV
* EXPOSE
* ONBUILD
* USER
* VOLIME
* WORKDIR

#### 使用格式

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

#### 命令选项

表 1‑4 docker import 命令选项

|  |  |  |
| --- | --- | --- |
| Name, shorthand | Default | Description |
| --change, -c [instruction] |  | 对创建的镜像执行 Dockerfile 指令 |
| --message, -m |  | 为导入的镜像设置 commit message |
| --platform |  | 如果服务器支持多平台特性使用该参数设置平台 |

#### 使用例子

注意，下面命令在执行期间为了保留文件的所有权（尤其是 root 的所有权），请使用 sudo 确保是 root 权限下执行该命令，否则文件所有权可能不会被保留。[4]

##### 从远程位置导入

$ docker import http://example.com/exampleimage.tgz

##### 从本地文件导入

* 通过管道使用 STDIN 导入

$ cat exampleimage.tgz | docker import - exampleimagelocal:new

* 导入时设置 commit message

$ cat exampleimage.tgz | docker import --message \

"New image imported from tarball" - exampleimagelocal:new

* 通过本地文件导入

$ docker import /path/to/exampleimage.tgz

##### 从本地文件目录导入

通过 tar -c 命令将目录中的文件都打包成一个 tar 文件，再通过 docker import 命令导入容器镜像：

$ sudo tar -c . | docker import - exampleimagedir

##### 导入时重新构建

$ sudo tar -c . | docker import --change "ENV DEBUG true" - exampleimagedir

## docker save 和 docker export 的区别[5]

* 在使用上，docker save 用于导出镜像（image），docker export 用于导出容器（container），两者针对的目标不一样；
* docker load 用于导入镜像压缩包，docker import 用于导入容器压缩包，两者虽然压缩包产生来源不同，但是最终都会导入成镜像（image）；
* docker load 不能对导入镜像进行重命名，而 docker import 则可以在通过 -（破折号）对导入的容器镜像指定新的名称。

## 镜像包和容器包的区别

上面介绍docker save 和 docker export 的区别时，可以看到两个命令虽然针对的数据来源不一样，但都是导出成 tar 压缩文件，而且最终都是导入 Docker 镜像。那么两者，导出的文件是一样的吗？如果导出的文件结构相似，应该可以用 docker load 命令导入 docker export 导出的文件，同时可以使用 docker docker import 命令导入 docker save 导出的文件。

为了验证这个猜测，我用 docker save 和 docker export 命令分别导出了一份 Nginx 镜像以及一份 Nginx 容器：

$ docker save -o **nginx-save.tar** nginx:latest

$ docker export -o **nginx-export.tar** nginx

首先看看两个 tar 文件的大小：

* 125M nginx-save.tar
* 123M nginx-export.tar

docker save 导出的 tar 压缩包稍微大一些。

### 交叉尝试导入命令

接着试试 docker load 是否可以导入 nginx-save.tar 压缩包：[5]

$ docker load -i nginx-export.tar

open /var/lib/docker/tmp/docker-import-304454769/bin/json: no such file or directory

很遗憾，docker load 并不能载入导出的容器压缩包。

那么，反过来呢？docker import 是否可以载入导出的镜像压缩包呢？[5]

$ cat nginx-save.tar | docker import - nginx:save

sha256:080616cf1d19c22dde9b560770287af76c35c250dc3d66525acc58cc6dae81e0

$ docker images | grep save

REPOSITORY TAG IMAGE ID CREATED SIZE

nginx save 080616cf1d19 59 seconds ago 130MB

额？！成功了！

成功了我们就启动一个容器试试：[5]

$ docker run nginx:save

docker: Error response from daemon: No command specified.

See 'docker run --help'.

然鹅，虽然能够成功地导入一个镜像，但是这个镜像并不能够使用。这是为什么呢？

### 镜像包和容器包区别

要搞清楚到底怎么回事，我们先看看镜像包和容器包有什么区别：[5]

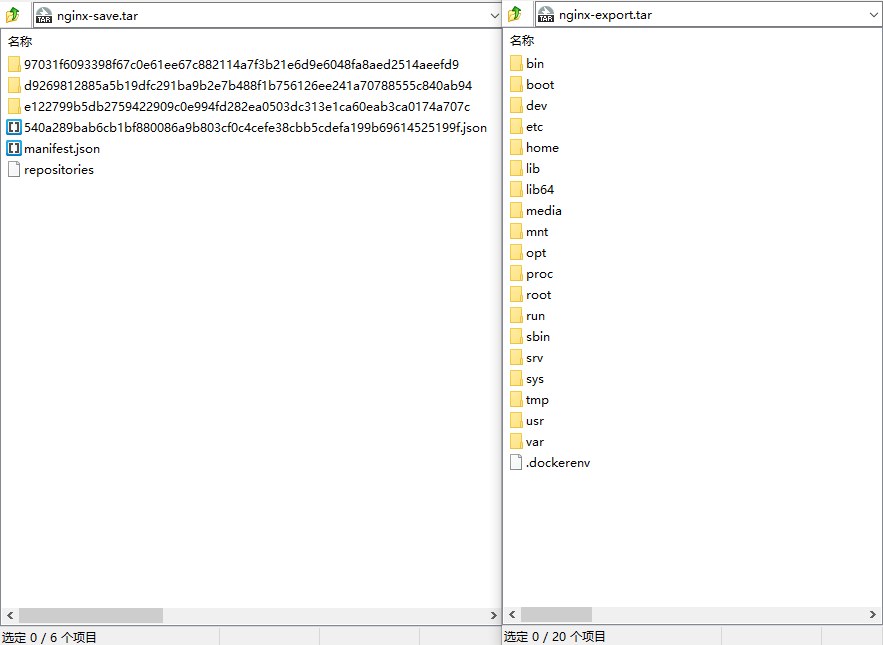


图 1‑5 docker save 和 docker export 导出文件目录结构

从上面的截图中可以看到，右边 nginx-export.tar 的目录结构是一个 Linux 系统的文件目录，结合 docker export 官方描述，猜测这是一个 Linux 文件系统。而左边的 nginx-save.tar 中到底是些什么内容呢？点开一个文件夹看看：

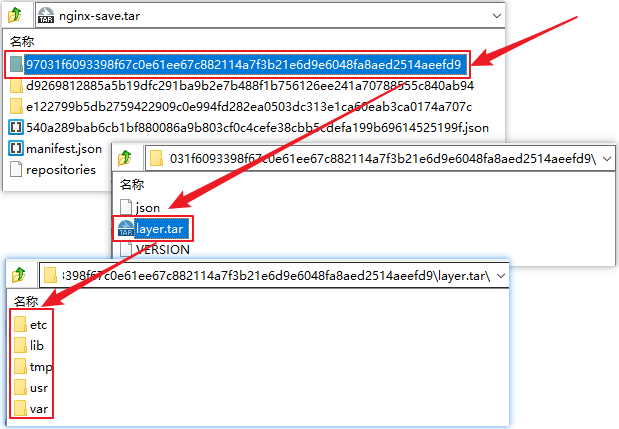


图 1‑6 docker save 导出文件内容结构

其实就是一个分层的文件系统，Docker 镜像实际上是由这样一层层（layers）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 nginx-save.tar 中各层文件叠加合并到一起，基本上就是 nginx-export.tar 的内容了。正式由于 nginx-save.tar 中的各层级文件可能包含着很多重复文件，所以 nginx-save.tar 文件大小会大于 nginx-export.tar。[5]

nginx-export.tar 把各层文件都叠加合并了，nginx-save.tar 则保留着各层文件信息，也就是说，两者虽然最终都可以被导入成为镜像，但是前者应该会丢失镜像历史信息（即每一层的 layer 信息）。下面，分别将两个 tar 导入，查看两个导入镜像的历史信息是否和符合预期：

$ docker load -i nginx-save.tar

$ cat nginx-export.tar | docker import - nginx:export

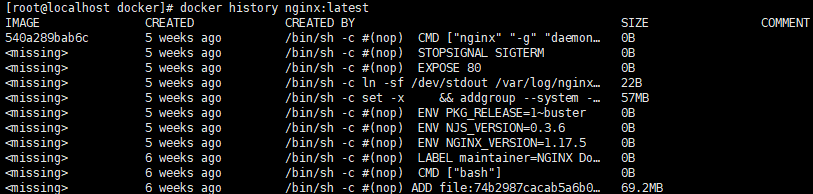


图 1‑7 nginx-save.tar 导入镜像历史信息



图 1‑8 nginx-export.tar 导入镜像历史信息

从上面截图中可以看出，docker export 导出的文件再 docker import 回去时，无法保留镜像所有历史信息，不能进行回滚操作。而 docker save 是依据镜像来导出的，所以导出时可以完整保留下每一层 layer 的信息。[6]

docker load 必须要载入一个分层的文件系统，而 nginx-export.tar 明显不具备这样的结构，因此无法载入。[5]

而 docker import 仅仅是将 tar 包里面的文件复制进来，不管 tar 包里面的文件结构是怎么样的，都可以载入进来，所以能够导入 nginx-save.tar。但是 nginx-save.tar 内部并不是一个有效的 Linux 文件系统，因此当我试图以该镜像启动容器时，容器并不能启动。[5]

### docker import 和 docker commit 的猜测[5]

首先来看看 docker import 命令的帮助信息：

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

-m, --message string Set commit message for imported image

似乎和 docker commit 的帮助信息很像：

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

-m, --message string Commit message

-p, --pause Pause container during commit (default true)

观察上述两个帮助信息，可以发现 docker import 和 docker commit 都有 –change 和 –message 选项。如果把 docker import 理解成时 docker commit 的进一步封装，那么过程可能是这样子的：

* 创建一个空白镜像；
* docker import 就是将外部文件系统复制进来；
* 然后使用 docker commit 将当前改动提交为一层文件系统，叠加到原有的镜像之上；
* 最后形成只有一层文件系统的镜像，也就是导入结果。

# 参考资料

1. 官方文档-[docker save](https://docs.docker.com/engine/reference/commandline/save/) [2019-12-01]
2. 官方文档-[docker load](https://docs.docker.com/engine/reference/commandline/load/) [2019-12-01]
3. 官方文档-[docker export](https://docs.docker.com/engine/reference/commandline/export/) [2019-12-01]
4. 官方文档-[docker import](https://docs.docker.com/engine/reference/commandline/import/) [2019-12-01]

1. [docker save与docker export的区别](https://jingsam.github.io/2017/08/26/docker-save-and-docker-export.html) - jingsam [2017-08-26]

1. [Docker镜像的导入导出](https://blog.csdn.net/ncdx111/article/details/79878098) - 小苏少 [2018-04-12]