Part 4.
Managing docker images

Docker images

A docker image is a snapshot of the filesystem + some metadata

- immutable
- copy-on-write storage
  - for instantiating containers
  - for creating new versions of the image (multiple layers)
- identified by unique hex IDs
  - `Image ID`: randomly genreated
  - `Digest`: hashed from the content
- may be tagged[10] with a human-friendly name
  eg: `debian:wheezy debian:jessie debian:latest`

[10] possibly multiple times

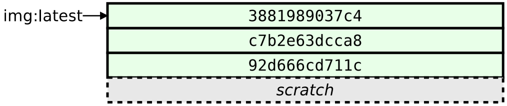
Image management commands

command	**description**
`docker images`	list all local images
`docker history` *image*	show the image history (list of ancestors)
`docker inspect` *image*...	show low-level infos (in json format)
`docker tag` *image tag*	tag an image
`docker commit` *container image*	create an image (from a container)
`docker import` *url*|- [*tag*]	create an image (from a tarball)
`docker rmi` *image*...	delete images

Example: images & containers

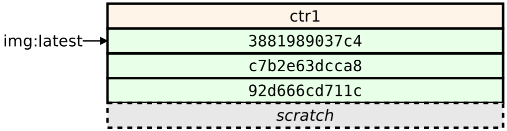
Example: images & containers

```
docker pull img
```



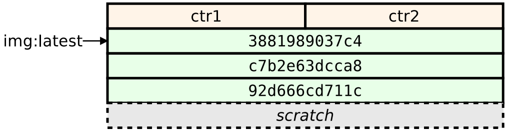
Example: images & containers

```
docker run --name ctr1 img
```



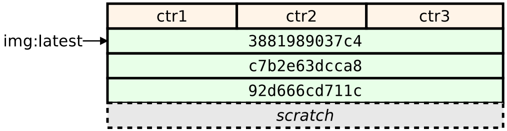
Example: images & containers

```
docker run --name ctr2 img
```



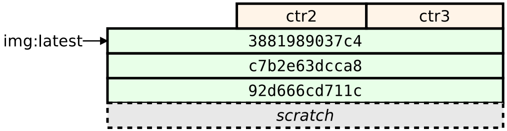
Example: images & containers

```
docker run --name ctr3 img
```



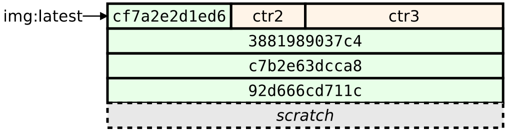
Example: images & containers

```
docker rm ctr1
```



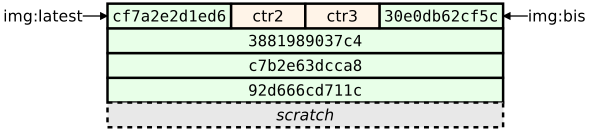
Example: images & containers

```
docker commit ctr2 img
```



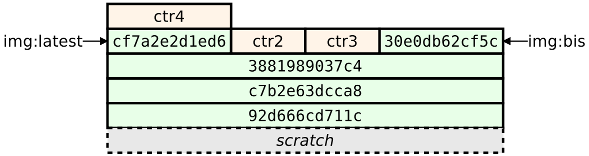
Example: images & containers

```
docker commit ctr3 img:bis
```



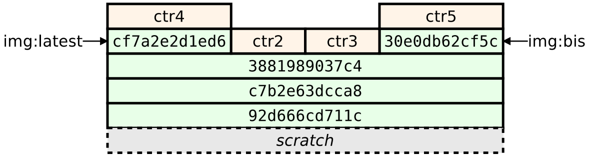
Example: images & containers

```
docker run --name ctr4 img
```



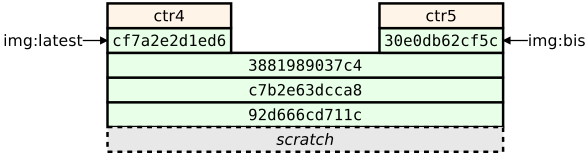
Example: images & containers

```
docker run --name ctr5 img:bis
```



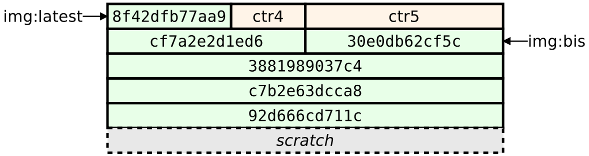
Example: images & containers

```
docker rm ctr2 ctr3
```



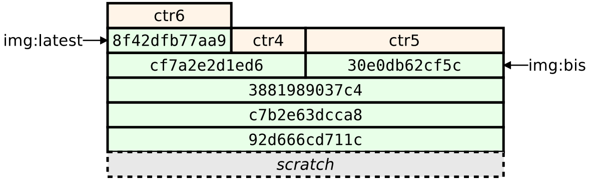
Example: images & containers

```
docker commit ctr4 img
```



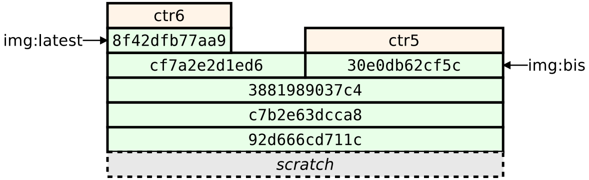
Example: images & containers

```
docker run --name ctr6 img
```



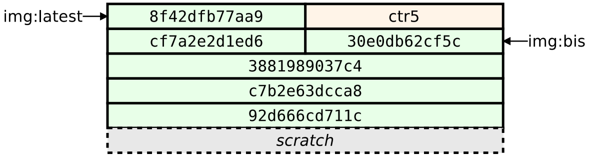
Example: images & containers

```
docker rm ctr4
```



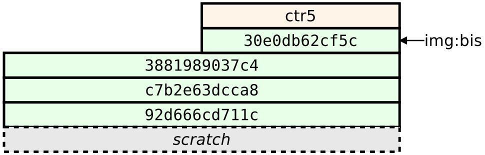
Example: images & containers

```
docker rm ctr6
```



Example: images & containers

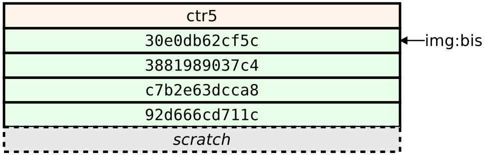
```
docker rmi img
```



Example: images & containers

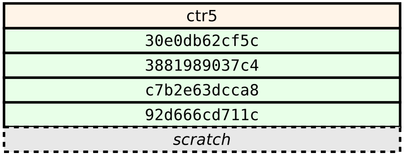
```
docker rmi img:bis
```

Error: image img:bis is reference by ctr5



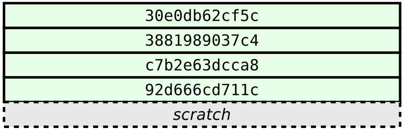
Example: images & containers

```
docker rmi -f img:bis
```



Example: images & containers

```
docker rm ctr5
```

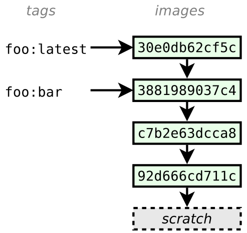


Example: images & containers

```
docker rmi 30e0
```

Images vs. Layers

docker < v1.10
no distinction between images & layers



docker >= v1.10
layers are hidden to the user
(implementation detail)

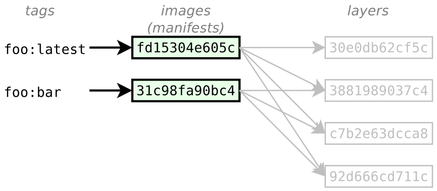


Image tags

A docker tag is made of two parts: *"REPOSITORY:TAG"*

The *TAG* part identifies the version of the image. If not provided, the default is "`:latest`"

```
$ docker images
REPOSITORY  TAG           IMAGE ID      CREATED        VIRTUAL SIZE
debian      8             835c4d274060  2 weeks ago    122.6 MB
debian      8.0           835c4d274060  2 weeks ago    122.6 MB
debian      jessie        835c4d274060  2 weeks ago    122.6 MB
debian      rc-buggy      350a74df81b1  7 months ago   159.9 MB
debian      experimental  36d6c9c7df4c  7 months ago   159.9 MB
debian      6.0.9         3b36e4176538  7 months ago   112.4 MB
debian      squeeze       3b36e4176538  7 months ago   112.4 MB
debian      wheezy        667250f9a437  7 months ago   115 MB
debian      latest        667250f9a437  7 months ago   115 MB
debian      7.5           667250f9a437  7 months ago   115 MB
debian      unstable      24a4621560e4  7 months ago   123.6 MB
debian      testing       7f5d8ca9fdcf  7 months ago   121.8 MB
debian      stable        caa04aa09d69  7 months ago   115 MB
debian      sid           f3d4759f77a7  7 months ago   123.6 MB
debian      7.4           e565fbbc6033  9 months ago   115 MB
debian      7.3           b5fe16f2ccba  11 months ago  117.8 MB
```

Tagging conventions (1/2)

Local tags may have arbitrary names, however the `docker push` and `docker pull` commands expect some conventions

The *REPOSITORY* identifies the origin of the image, it may be:

- a name (eg: `debian`)
  → refers to a repository on the official registry
  → `https://store.docker.com/`
- a hostname+name (eg: `some.server.com/repo`)
  → refers to an arbitrary server supporting the registry API
  → `https://docs.docker.com/reference/api/registry_api/`

Tagging conventions (2/2)

Use slashes to delimit namespaces (for subprojects):

image name	**description**
`debian`	(semi-)official debian images
`fedora`	official fedora images
`fedora/apache`	apache images provided by the fedora project
`fedora/couchdb`	couchdb images provided by the fedora project

Image transfer commands

Using the registry API

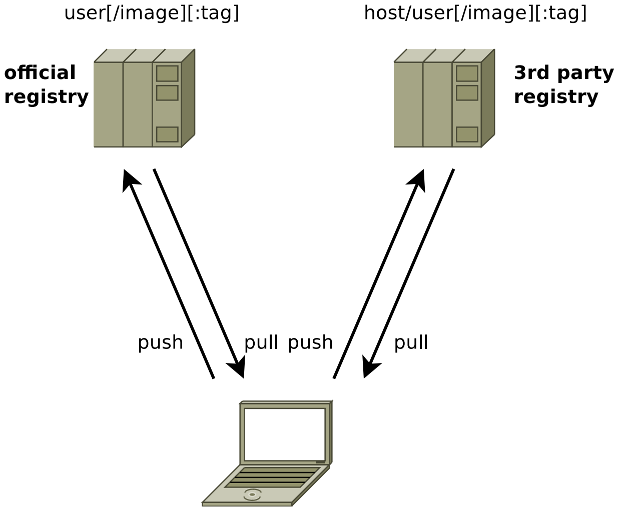
`docker pull` *repo*[:*tag*]. . .	pull an image/repo from a registry
`docker push` *repo*[:*tag*]. . .	push an image/repo from a registry
`docker search` *text*	search an image on the official registry
`docker login` . . .	login to a registry
`docker logout` . . .	logout from a registry

Manual transfer

`docker save` *repo*[:*tag*]. . .	export an image/repo as a tarball
`docker load`	load images from a tarball
`docker-ssh`[11] . . .	proposed script to transfer images between two daemons over ssh

[11] `https://github.com/a-ba/docker-utils/`

Transferring images


user[/image][:tag]
host/user[/image][:tag]
official registry
3rd party registry
push
pull
push
pull