docker的子命令

查看docker的子命令，直接敲docker或完整的docker help就可以了:

root@tankywoo-docker:~# docker [1/1617]

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 a container 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

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

version Show the docker version information

wait Block until a container stops, then print its exit code

查看docker支持的选项:

docker --help

**常用命令**

总结一下常用命令:

其中<>阔起来的参数为必选，[]阔起来为可选

* docker version 查看docker的版本号，包括客户端、服务端、依赖的Go等
* docker info 查看系统(docker)层面信息，包括管理的images, containers数等
* docker search <image> 在docker index中搜索image
* docker pull <image> 从docker registry server 中下拉image
* docker push <image|repository> 推送一个image或repository到registry
* docker push <image|repository>:TAG 同上，指定tag
* docker inspect <image|container> 查看image或container的底层信息
* docker images **TODO** filter out the intermediate image layers (intermediate image layers 是什么)
* docker images -a 列出所有的images
* docker ps 默认显示正在运行中的container
* docker ps -l 显示最后一次创建的container，包括未运行的
* docker ps -a 显示所有的container，包括未运行的
* docker logs <container> 查看container的日志，也就是执行命令的一些输出
* docker rm <container...> 删除一个或多个container
* docker rm `docker ps -a -q` 删除所有的container
* docker ps -a -q | xargs docker rm 同上, 删除所有的container
* docker rmi <image...> 删除一个或多个image
* docker start/stop/restart <container> 开启/停止/重启container
* docker start -i <container> 启动一个container并进入交互模式
* docker attach <container> attach一个运行中的container
* docker run <image> <command> 使用image创建container并执行相应命令，然后停止
* docker run -i -t <image> /bin/bash 使用image创建container并进入交互模式, login shell是/bin/bash
* docker run -i -t -p <host\_port:contain\_port> 将container的端口映射到宿主机的端口
* docker commit <container> [repo:tag] 将一个container固化为一个新的image，后面的repo:tag可选
* docker build <path> 寻找path路径下名为的Dockerfile的配置文件，使用此配置生成新的image
* docker build -t repo[:tag] 同上，可以指定repo和可选的tag
* docker build - < <dockerfile> 使用指定的dockerfile配置文件，docker以stdin方式获取内容，使用此配置生成新的image
* docker port <container> <container port> 查看本地哪个端口映射到container的指定端口，其实用docker ps 也可以看到

**使用images新建一个container并登录**

使用image来创建container:

root@tankywoo-docker:~# docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

ubuntu 13.10 5e019ab7bf6d 12 days ago 180 MB

ubuntu saucy 5e019ab7bf6d 12 days ago 180 MB

ubuntu 12.04 74fe38d11401 12 days ago 209.6 MB

ubuntu precise 74fe38d11401 12 days ago 209.6 MB

root@tankywoo-docker:~# docker run -i -t 74fe38d11401 /bin/bash

root@80c761d06a87:/# cat /etc/issue

Ubuntu 12.04.4 LTS \n \l

使用repository来创建container, 这时默认使用tag为lastest的image:

root@tankywoo-docker:~# docker run -i -t ubuntu /bin/bash

root@442e1cc85a8d:/# uname -a

Linux 442e1cc85a8d 3.8.0-25-generic #37~precise1-Ubuntu SMP Fri Jun 7 16:27:35 UTC 2013 x86\_64 x86\_64 x86\_64 GNU/Linux

root@442e1cc85a8d:/# cat /etc/issue

Ubuntu 14.04 LTS \n \l

root@442e1cc85a8d:/# exit

**使用commit将一个container固化为一个image**

root@tankywoo-docker:~# docker ps -l

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

f1fd375204af ubuntu:12.04 /bin/bash 10 minutes ago Exited (127) 48 seconds ago lonely\_colden

root@tankywoo-docker:~# docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

ubuntu 13.10 5e019ab7bf6d 12 days ago 180 MB

ubuntu saucy 5e019ab7bf6d 12 days ago 180 MB

ubuntu 12.04 74fe38d11401 12 days ago 209.6 MB

提交当前container为一个image，顺便带上作者信息，并指定repository 和 tag

root@tankywoo-docker:~# docker commit -a "Tanky Woo <me@tankywoo.com>" f1fd375204af ubuntu:test

fe65a2781daea01c67c33f11868abe6d510833bca07b90fc681cdfe98a9196ac

root@tankywoo-docker:~# docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

ubuntu test fe65a2781dae 6 seconds ago 209.6 MB

ubuntu 13.10 5e019ab7bf6d 12 days ago 180 MB

ubuntu saucy 5e019ab7bf6d 12 days ago 180 MB

**attach一个运行中的容器**

root@tankywoo-docker:~# docker ps -l

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

e2e6c95f0bf5 ubuntu:test /bin/bash 11 minutes ago Exited (0) 11 minutes ago suspicious\_mccarthy

启动一个container:

root@tankywoo-docker:~# docker start e2e6c95f0bf5

e2e6c95f0bf5

可以看此container到正在运行中:

root@tankywoo-docker:~# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

e2e6c95f0bf5 ubuntu:test /bin/bash 11 minutes ago Up 2 seconds suspicious\_mccarthy

attach这个container:

root@tankywoo-docker:~# docker attach e2e6c95f0bf5

进入container:

root@e2e6c95f0bf5:/#

**docker build 构建**

root@tankywoo-docker:~# cat Dockerfile

FROM ubuntu:test

ENTRYPOINT echo "Welcome!"

root@tankywoo-docker:~# docker build -t ubuntu:newtest - < Dockerfile

Uploading context 2.048 kB

Uploading context

Step 0 : FROM ubuntu:test

---> fe65a2781dae

Step 1 : ENTRYPOINT echo "Welcome!"

---> Running in 09a062a296c5

---> f8104f05df90

Successfully built f8104f05df90

Removing intermediate container 09a062a296c5

root@tankywoo-docker:~# docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

ubuntu newtest f8104f05df90 8 seconds ago 209.6 MB

ubuntu test fe65a2781dae 23 minutes ago 209.6 MB

ubuntu 13.10 5e019ab7bf6d 12 days ago 180 MB

ubuntu saucy 5e019ab7bf6d 12 days ago 180 MB

ubuntu precise 74fe38d11401 12 days ago 209.6 MB

ubuntu 12.04 74fe38d11401 12 days ago 209.6 MB

ubuntu 12.10 a7cf8ae4e998 12 days ago 171.3 MB

ubuntu quantal a7cf8ae4e998 12 days ago 171.3 MB

ubuntu 14.04 99ec81b80c55 12 days ago 266 MB

ubuntu trusty 99ec81b80c55 12 days ago 266 MB

ubuntu latest 99ec81b80c55 12 days ago 266 MB

ubuntu 13.04 316b678ddf48 12 days ago 169.4 MB

ubuntu raring 316b678ddf48 12 days ago 169.4 MB

ubuntu 10.04 3db9c44f4520 2 weeks ago 183 MB

ubuntu lucid 3db9c44f4520 2 weeks ago 183 MB

root@tankywoo-docker:~# docker run ubuntu:newtest

2014/05/07 17:30:34 Unrecognized input header

root@tankywoo-docker:~# docker run -i -t ubuntu:newtest /bin/bash

Welcome!

TODO: 为何要使用 -i 和 -t

**使用 docker run -p 的例子**

镜像ubuntu:12.04没有vi，没法编辑/etc/apt/sources.list

现在本地有一份，想上传上去

首先映射端口(宿主的2222端口和container的33333端口映射):

docker run -i -t -p 22222:33333 fe65a2781dae /bin/bash

container上监听33333：

nc -l -p 33333 > /etc/apt/sources.list

本地使用22222端口传输:

nc localhost 22222 < sources.list

**查看映射的端口**

root@tankywoo-docker:~# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

7abe8e31ac8b ubuntu:test /bin/bash 15 minutes ago Up 15 minutes 0.0.0.0:22222->33333/tcp hungry\_carson

root@tankywoo-docker:~# docker port 7abe8e31ac8b 33333

0.0.0.0:22222

root@tankywoo-docker:~# netstat -tlnp

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name

tcp 0 0 0.0.0.0:22 0.0.0.0:\* LISTEN 528/sshd

tcp6 0 0 :::22222 :::\* LISTEN 12946/docker

tcp6 0 0 :::22 :::\* LISTEN 528/sshd

但是这里很好奇为啥是监听在ipv4的地址上?

**删除image/container遇到的依赖关系**

关于删除时的依赖关系，按照提示删除就行了

比如删除images时，需要先删除通过它创建的所有containers:

root@tankywoo-docker:~# docker rmi 666c5d65f396 3494872e31a4 62fda5e450d5 5e1829f90d6e 89554a25c998

Error: Conflict, cannot delete 666c5d65f396 because the container 43a7072bac7a is using it

Error: Conflict, cannot delete 3494872e31a4 because the container 40b3cd8b2e42 is using it

Error: Conflict, cannot delete 62fda5e450d5 because the container 5142a3d092a6 is using it

Untagged: test:latest

Deleted: 5e1829f90d6e9ac09645841fe6ab85a0b0f9b28f008a571299a624e566684afe

Deleted: ae5ae236a8e1d946963a7c2c142cc892b1979cb9458e0ecac4d33d2283ace567

Untagged: memchaced:latest

Deleted: 89554a25c998d14c76ff885ddac7cc1a47ae4caf9edcddaa43408b402a1684fb

2014/05/07 15:44:41 Error: failed to remove one or more images

root@tankywoo-docker:~# docker rm 43a7072bac7a 40b3cd8b2e42 5142a3d092a6

43a7072bac7a

40b3cd8b2e42

5142a3d092a6

且删除images时也可能会遇到依赖其它的images，比如直接删除父镜像时，就会提示需要先删除子镜像。

可以通过:

docker images --tree

来查看，不过官方提示 --tree 已经弃用了，会在以后的版本去掉.

首先清空所有containers:

root@tankywoo-docker:~# docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

然后以树形结构查看依赖关系:

root@tankywoo-docker:~# docker images --tree

Warning: '--tree' is deprecated, it will be removed soon. See usage.

└─511136ea3c5a Virtual Size: 0 B

├─e2aa6665d371 Virtual Size: 106.1 MB

│ └─f0ee64c4df74 Virtual Size: 106.3 MB

│ └─2209cbf9dcd3 Virtual Size: 106.3 MB

│ └─5e019ab7bf6d Virtual Size: 180 MB Tags: ubuntu:13.10, ubuntu:saucy

├─f10ebce2c0e1 Virtual Size: 103.7 MB

│ └─82cdea7ab5b5 Virtual Size: 103.9 MB

│ └─5dbd9cb5a02f Virtual Size: 103.9 MB

│ └─74fe38d11401 Virtual Size: 209.6 MB Tags: ubuntu:precise, ubuntu:12.04

│ └─fe65a2781dae Virtual Size: 209.6 MB Tags: ubuntu:test

│ └─276cc641e40e Virtual Size: 388.3 MB Tags: ubuntu:newtest

├─ef519c9ee91a Virtual Size: 100.9 MB

│ └─07302703becc Virtual Size: 101.2 MB

│ └─cf8dc907452c Virtual Size: 101.2 MB

│ └─a7cf8ae4e998 Virtual Size: 171.3 MB Tags: ubuntu:12.10, ubuntu:quantal

├─5e66087f3ffe Virtual Size: 192.5 MB

│ └─4d26dd3ebc1c Virtual Size: 192.7 MB

│ └─d4010efcfd86 Virtual Size: 192.7 MB

│ └─99ec81b80c55 Virtual Size: 266 MB Tags: ubuntu:14.04, ubuntu:latest, ubuntu:trusty

├─02dae1c13f51 Virtual Size: 98.35 MB

│ └─e7206bfc66aa Virtual Size: 98.54 MB

│ └─cb12405ee8fa Virtual Size: 98.54 MB

│ └─316b678ddf48 Virtual Size: 169.4 MB Tags: ubuntu:raring, ubuntu:13.04

└─6cfa4d1f33fb Virtual Size: 0 B

└─3db9c44f4520 Virtual Size: 183 MB Tags: ubuntu:10.04, ubuntu:lucid

现在准备删除12.10版本的父镜像 cf8dc907452c, 会提示有冲突，删不掉:

root@tankywoo-docker:~# docker rmi cf8dc907452c

Error: Conflict, cf8dc907452c wasn't deleted

2014/05/07 18:49:35 Error: failed to remove one or more images

但是可以删除叶子节点 a7cf8ae4e998:

root@tankywoo-docker:~# docker rmi a7cf8ae4e998

Untagged: ubuntu:12.10

Untagged: ubuntu:quantal

Deleted: a7cf8ae4e998c5339e769d6cc466f9133bd4d330a549bb846cb1641cd638247c

Deleted: cf8dc907452c970224551599da573c9e32897fc65286d942625c4c86dabd680d

Deleted: 07302703beccc2ea25f34333decad32ed06446e8a14c020ffbd0be017364b9fe

Deleted: ef519c9ee91a06fc33cefbda1bce27686617761700252dff0397f2c0e269f3c5

**containers之间共享数据**

docker 的 containers之间共享目录是通过 volume 。

docker run 命令使用 -v 可以绑定一个volume, -v 可以使用多次，创建多个volume:

root@tankywoo-docker:~# docker run -i -t -v /tmp/tankywoo --name data ubuntu:newtest /bin/bash [6/3516]

使用 mount 看到 /tmp/tankywoo 已经被mount了:

root@fec65f523cef:/# mount

none on / type aufs (rw,relatime,si=f7ac8b1595d13ed9)

...

/dev/disk/by-uuid/b77aed99-bb9b-4881-9702-4ed204fe5d46 on /tmp/tankywoo type ext3 (rw,relatime,errors=remount-ro,user\_xattr,acl,barrier=1,data=ordered)

查看 /tmp/tankywoo 目录下，是空的:

root@fec65f523cef:/tmp/tankywoo# ls

root@fec65f523cef:/tmp/tankywoo#

然后在宿主机新建一个container，来绑定这个volume:

按照 docker run 的命令行参数:

--volumes-from=[]: Mount volumes from the specified container(s)

有问题:

root@tankywoo-docker:/tmp/tankywoo# docker run -i -t --volumes-from=["data"] ubuntu:newtest /bin/bash [21/158]

2014/05/08 15:58:19 Error: Cannot start container 5d83dcaf8f0220024e0403a362c0512a8218cfcb45dc911df5d2cd37f9a4e8a4: Container [data] not found. Impossible to mount its volumes

必须像short option的方式使用:

root@tankywoo-docker:/tmp/tankywoo# docker run -i -t --volumes-from data ubuntu:newtest /bin/bash

root@d100d9604b4b:/# mount

none on / type aufs (rw,relatime,si=f7ac8b15b25036d9)

...

/dev/disk/by-uuid/b77aed99-bb9b-4881-9702-4ed204fe5d46 on /tmp/tankywoo type ext3 (rw,relatime,errors=remount-ro,user\_xattr,acl,barrier=1,data=ordered)

也可以看到 /tmp/tankywoo 目录，并且是空的，然后新建一个文件:

root@d100d9604b4b:/tmp/tankywoo# ls

root@d100d9604b4b:/tmp/tankywoo# touch file

root@d100d9604b4b:/tmp/tankywoo# ls

file

再看看之前那个container:

root@fec65f523cef:/tmp/tankywoo# ls

file

也有这个文件了

参考

**退出container但是保持运行**

默认情况下，如果使用ctrl-c退出container,那么container也会stop

按ctrl-p ctrl-q可以退出到宿主机，而保持container仍然在运行

**Docker被墙**

关于 Docker 被墙，老甘的文章里提到的修改hosts文件，先mark，未验证:

# /etc/hosts

54.234.135.251 get.docker.io

54.234.135.251 cdn-registry-1.docker.io

**遗留的问题**

有时docker执行不了任何命令(会卡住)，包括重启docker server，在日志里看到这些:

May 5 17:41:48 tpl-ubuntu12-04 kernel: [99589.489241] unregister\_netdevice: waiting for lo to become free. Usage count = 3

May 5 17:41:58 tpl-ubuntu12-04 kernel: [99599.708117] unregister\_netdevice: waiting for lo to become free. Usage count = 3

May 5 17:42:08 tpl-ubuntu12-04 kernel: [99609.927057] unregister\_netdevice: waiting for lo to become free. Usage count = 3

May 5 17:42:18 tpl-ubuntu12-04 kernel: [99620.145993] unregister\_netdevice: waiting for lo to become free. Usage count = 3

May 5 17:42:29 tpl-ubuntu12-04 kernel: [99630.364922] unregister\_netdevice: waiting for lo to become free. Usage count = 3

May 5 17:42:39 tpl-ubuntu12-04 kernel: [99640.583850] unregister\_netdevice: waiting for lo to become free. Usage count = 3

May 5 17:42:49 tpl-ubuntu12-04 kernel: [99650.802794] unregister\_netdevice: waiting for lo to become free. Usage count = 3

May 5 17:42:59 tpl-ubuntu12-04 kernel: [99661.021726] unregister\_netdevice: waiting for lo to become free. Usage count = 3

May 5 17:43:10 tpl-ubuntu12-04 kernel: [99671.240662] unregister\_netdevice: waiting for lo to become free. Usage count = 3

May 5 17:43:20 tpl-ubuntu12-04 kernel: [99681.459572] unregister\_netdevice: waiting for lo to become free. Usage count = 3

May 5 17:43:30 tpl-ubuntu12-04 kernel: [99691.678530] unregister\_netdevice: waiting for lo to become free. Usage count = 3

May 5 17:43:40 tpl-ubuntu12-04 kernel: [99701.897432] unregister\_netdevice: waiting for lo to become free. Usage count = 3

May 5 17:43:51 tpl-ubuntu12-04 kernel: [99712.128370] unregister\_netdevice: waiting for lo to become free. Usage count = 3

May 5 17:44:01 tpl-ubuntu12-04 kernel: [99722.347289] unregister\_netdevice: waiting for lo to become free. Usage count = 3

May 5 17:44:11 tpl-ubuntu12-04 kernel: [99732.566226] unregister\_netdevice: waiting for lo to become free. Usage count = 3

May 5 17:44:21 tpl-ubuntu12-04 kernel: [99742.785141] unregister\_netdevice: waiting for lo to become free. Usage count = 3

**什么是Layer**

Docker images are built up in layers. So, for instance, if you need to run WordPress, you would build the Ubuntu layer, add a layer for Apache2 web server, add a PHP layer and then a layer for the WordPress files. Lower layers can be re-used. We might take the PHP layer and layer on Drupal instead of WordPress, or update our WordPress layer with a newer version or Wordpress.

Because we can re-use layers, we can make new docker images very cheaply. We can create a new docker image by changing just a single line of one file and we do not have to rebuild the whole stack.

The beauty of docker images being “just files” means that the difference between two docker images is just a diff of the files they contain.

[Hykes Explains Docker](http://www.activestate.com/blog/2013/06/solomon-hykes-explains-docker)

**概念上的问题**

[The Docker Guidebook](http://kencochrane.net/blog/2013/08/the-docker-guidebook/) 的简单对比:

Image : An image is a read only layer used to build a container. They do not change.

Container : Is basically a self contained runtime environment that is built using one or more images. You can commit your changes to a container and create an image.

index / registry : These are public or private servers where people can upload their repositories so they can easily share what they made.

Repository : A repository is a group of images located in the docker registry. There are two types of repositories, Top level and user repositories. Top level repositories don't have a '/' in the name and they are usually reserved for base images. These Top level repositories is what most people build their repositories on top of. They are controlled by the maintainers of Docker. User repositories are repositories that anyone can upload into the registry and share with other people.

说直接点，Image和Container最容易理解和对比，它俩的关系就像类与类的实例这两的关系一样。

其实Index和Registry也有区别，主要就是Index存储的是用户信息、images的checksum；而Registry存储的是images。具体见官方文档[Registry & Index Spec](http://docs.docker.io/reference/api/registry_index_spec/)。

另外，关于Repository与Registry和Image又是什么关系?

root@tankywoo-docker:~/docker-registry-master# docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

10.2.15.190/tankywoo latest 276cc641e40e 4 days ago 388.3 MB

10.2.15.190:5000/tankywoo latest 276cc641e40e 4 days ago 388.3 MB

ubuntu newtest 276cc641e40e 4 days ago 388.3 MB

ubuntu test fe65a2781dae 4 days ago 209.6 MB

ubuntu 13.10 5e019ab7bf6d 2 weeks ago 180 MB

ubuntu saucy 5e019ab7bf6d 2 weeks ago 180 MB

ubuntu 12.04 74fe38d11401 2 weeks ago 209.6 MB

ubuntu precise 74fe38d11401 2 weeks ago 209.6 MB

ubuntu 14.04 99ec81b80c55 2 weeks ago 266 MB

ubuntu latest 99ec81b80c55 2 weeks ago 266 MB

ubuntu trusty 99ec81b80c55 2 weeks ago 266 MB

ubuntu 13.04 316b678ddf48 2 weeks ago 169.4 MB

ubuntu raring 316b678ddf48 2 weeks ago 169.4 MB

busybox latest 2d8e5b282c81 2 weeks ago 2.489 MB

ubuntu 10.04 3db9c44f4520 2 weeks ago 183 MB

ubuntu lucid 3db9c44f4520 2 weeks ago 183 MB

以这个为例

这里的ubuntu是image名称吗？(后面解答)

一个image完整的名称是:

username/image\_name:tag

docker整体和Github非常像，image管理也不例外。

其中，如果username没有写，则被认为是官方认证过的image。如前面提到，如果tag没有写，则被认为tag是lastest。

另外，如果username写了，如 tankywoo/ubuntu，则会在官方index中查找username为tankywoo的ubuntu仓库；如果写的如上10.2.15.190:5000/tankywoo，则10.2.15.190:5000则被认为是第三方registry的地址。

所以如上所说，ubuntu并不是image的名称，而是repository的名称。

再看看/var/lib/docker/ 下的 repositories-aufs，这是一个repositories的json列表:

root@tankywoo-docker:~/docker-registry-master# cat /var/lib/docker/repositories-aufs | python -m json.tool

{

"Repositories": {

"10.2.15.190/tankywoo": {

"latest": "276cc641e40e01a18f6bee9e81a576adb7090d3fbae098f809857e0696ccbc87"

},

"10.2.15.190:5000/tankywoo": {

"latest": "276cc641e40e01a18f6bee9e81a576adb7090d3fbae098f809857e0696ccbc87"

},

"busybox": {

"latest": "2d8e5b282c81244037eb15b2068e1c46319c1a42b80493acb128da24b2090739"

},

"ubuntu": {

"10.04": "3db9c44f45209632d6050b35958829c3a2aa256d81b9a7be45b362ff85c54710",

"12.04": "74fe38d114018aac73c5997b95263090048ec9a1f58f33a1b53f55e92156d53b",

"13.04": "316b678ddf487a37012630ae3219c8bb78c1f4b58d31c9513c3ea6b88f9e5635",

"13.10": "5e019ab7bf6deb75b211411ef7257d1e76bf7edee31d9da62a392df98d0529d6",

"14.04": "99ec81b80c55d906afd8179560fdab0ee93e32c52053816ca1d531597c1ff48f",

"latest": "99ec81b80c55d906afd8179560fdab0ee93e32c52053816ca1d531597c1ff48f",

"lucid": "3db9c44f45209632d6050b35958829c3a2aa256d81b9a7be45b362ff85c54710",

"newtest": "276cc641e40e01a18f6bee9e81a576adb7090d3fbae098f809857e0696ccbc87",

"precise": "74fe38d114018aac73c5997b95263090048ec9a1f58f33a1b53f55e92156d53b",

"raring": "316b678ddf487a37012630ae3219c8bb78c1f4b58d31c9513c3ea6b88f9e5635",

"saucy": "5e019ab7bf6deb75b211411ef7257d1e76bf7edee31d9da62a392df98d0529d6",

"test": "fe65a2781daea01c67c33f11868abe6d510833bca07b90fc681cdfe98a9196ac",

"trusty": "99ec81b80c55d906afd8179560fdab0ee93e32c52053816ca1d531597c1ff48f"

}

}

}

可以看到 10.2.15.190/tankywoo, 10.2.15.190:5000/tankywoo, busybox, ubuntu 等都是repository名，里面包含了一个或多个images。

* registry是repositories的集合
* repositry是images的集合