title: Docker(五)仓库 tags: Docker categories: 技术

Docker Registry

镜像构建完成后，可以很容易的在当前宿主上运行，但是，如果需要在其它服务器上使用这个镜像，我们就需要一个集中的存储、分发镜像的服务，Docker Registry 就是这样的服务。

一个 **Docker Registry** 中可以包含多个**仓库**（Repository）；每个仓库可以包含多个**标签**（Tag）；每个标签对应一个镜像。

通常，一个仓库会包含同一个软件不同版本的镜像，而标签就常用于对应该软件的各个版本。我们可以通过 <仓库名>:<标签> 的格式来指定具体是这个软件哪个版本的镜像。如果不给出标签，将以 latest 作为默认标签。

以 Ubuntu 镜像 为例，ubuntu 是仓库的名字，其内包含有不同的版本标签，如，14.04, 16.04。我们可以通过 ubuntu:14.04，或者 ubuntu:16.04 来具体指定所需哪个版本的镜像。如果忽略了标签，比如 ubuntu，那将视为 ubuntu:latest。

仓库名经常以 *两段式路径* 形式出现，比如 jwilder/nginx-proxy，前者往往意味着 Docker Registry 多用户环境下的用户名，后者则往往是对应的软件名。但这并非绝对，取决于所使用的具体 Docker Registry 的软件或服务。

Docker Registry 公开服务

Docker Registry 公开服务是开放给用户使用、允许用户管理镜像的 Registry 服务。一般这类公开服务允许用户免费上传、下载公开的镜像，并可能提供收费服务供用户管理私有镜像。

最常使用的 Registry 公开服务是官方的 Docker Hub，这也是默认的 Registry，并拥有大量的高质量的官方镜像。除此以外，还有 CoreOS 的 Quay.io，CoreOS 相关的镜像存储在这里；Google 的 Google Container Registry，Kubernetes 的镜像使用的就是这个服务。

由于某些原因，在国内访问这些服务可能会比较慢。国内的一些云服务商提供了针对 Docker Hub 的镜像服务（Registry Mirror），这些镜像服务被称为**加速器**。常见的有 阿里云加速器、DaoCloud 加速器、灵雀云加速器等。使用加速器会直接从国内的地址下载 Docker Hub 的镜像，比直接从官方网站下载速度会提高很多。在后面的章节中会有进一步如何配置加速器的讲解。

国内也有一些云服务商提供类似于 Docker Hub 的公开服务。比如 时速云镜像仓库、网易云镜像服务、DaoCloud 镜像市场、阿里云镜像库等。

私有 Docker Registry

除了使用公开服务外，用户还可以在本地搭建私有 Docker Registry。Docker 官方提供了 Docker Registry 镜像，可以直接使用做为私有 Registry 服务。在下面会进一步的进行搭建私有 Registry 服务的讲解。

开源的 Docker Registry 镜像只提供了 Docker Registry API 的服务端实现，足以支持 docker 命令，不影响使用。但不包含图形界面，以及镜像维护、用户管理、访问控制等高级功能。在官方的商业化版本 Docker Trusted Registry 中，提供了这些高级功能。

除了官方的 Docker Registry 外，还有第三方软件实现了 Docker Registry API，甚至提供了用户界面以及一些高级功能。比如，VMWare Harbor 和 Sonatype Nexus。

访问仓库

仓库（Repository）是集中存放镜像的地方。

一个容易混淆的概念是注册服务器（Registry）。实际上注册服务器是管理仓库的具体服务器，每个服务器上可以有多个仓库，而每个仓库下面有多个镜像。从这方面来说，仓库可以被认为是一个具体的项目或目录。例如对于仓库地址 dl.dockerpool.com/ubuntu 来说，dl.dockerpool.com 是注册服务器地址，ubuntu 是仓库名。

大部分时候，并不需要严格区分这两者的概念。

Docker Hub

目前 Docker 官方维护了一个公共仓库 Docker Hub，其中已经包括了超过 15,000 的镜像。大部分需求，都可以通过在 Docker Hub 中直接下载镜像来实现。

登录

可以通过执行 docker login 命令来输入用户名、密码和邮箱来完成注册和登录。 注册成功后，本地用户目录的 .dockercfg 中将保存用户的认证信息。

基本操作

用户无需登录即可通过 docker search 命令来查找官方仓库中的镜像，并利用 docker pull 命令来将它下载到本地。

例如以 centos 为关键词进行搜索：

```
$ sudo docker search centos
NAME                                              DESCRIPTION
STARS     OFFICIAL   AUTOMATED
centos                                            The official build of CentOS.
465       [OK]
tianon/centos                                     CentOS 5 and 6, created using
rinse instea...   28
blalor/centos                                     Bare-bones base CentOS 6.5
image                6                    [OK]
saltstack/centos-6-minimal
6                    [OK]
tutum/centos-6.4                                  DEPRECATED. Use
tutum/centos:6.4 instead. ...   5                      [OK]
...
```

可以看到返回了很多包含关键字的镜像，其中包括镜像名字、描述、星级（表示该镜像的受欢迎程度）、是否官方创建、是否自动创建。 官方的镜像说明是官方项目组创建和维护的，automated 资源允许用户验证镜像的

来源和内容。

根据是否是官方提供，可将镜像资源分为两类。 一种是类似 centos 这样的基础镜像，被称为基础或根镜像。这些基础镜像是由 Docker 公司创建、验证、支持、提供。这样的镜像往往使用单个单词作为名字。 还有一种类型，比如 tianon/centos 镜像，它是由 Docker 的用户创建并维护的，往往带有用户名称前缀。可以通过前缀 user_name/ 来指定使用某个用户提供的镜像，比如 tianon 用户。

另外，在查找的时候通过 -s N 参数可以指定仅显示评价为 N 星以上的镜像（新版本Docker推荐使用--filter=stars=N参数）。

下载官方 centos 镜像到本地。

```
$ sudo docker pull centos
Pulling repository centos
0b443ba03958: Download complete
539c0211cd76: Download complete
511136ea3c5a: Download complete
7064731afe90: Download complete
```

用户也可以在登录后通过 docker push 命令来将镜像推送到 Docker Hub。

自动创建

自动创建（Automated Builds）功能对于需要经常升级镜像内程序来说，十分方便。 有时候，用户创建了镜像，安装了某个软件，如果软件发布新版本则需要手动更新镜像。。

而自动创建允许用户通过 Docker Hub 指定跟踪一个目标网站（目前支持 GitHub 或 BitBucket）上的项目，一旦项目发生新的提交，则自动执行创建。

要配置自动创建，包括如下的步骤：

- 创建并登录 Docker Hub，以及目标网站；
- 在目标网站中连接帐户到 Docker Hub；
- 在 Docker Hub 中 配置一个自动创建；
- 选取一个目标网站中的项目（需要含 Dockerfile）和分支；
- 指定 Dockerfile 的位置，并提交创建。

之后，可以 在Docker Hub 的 自动创建页面 中跟踪每次创建的状态。

私有仓库

有时候使用 Docker Hub 这样的公共仓库可能不方便，用户可以创建一个本地仓库供私人使用。

docker-registry 是官方提供的工具，可以用于构建私有的镜像仓库。

安装运行 docker-registry

容器运行

在安装了 Docker 后，可以通过获取官方 registry 镜像来运行。

```
$ sudo docker run -d -p 5000:5000 registry
```

这将使用官方的 registry 镜像来启动本地的私有仓库。 用户可以通过指定参数来配置私有仓库位置，例如配置镜像存储到 Amazon S3 服务。

```
$ sudo docker run \
         -e SETTINGS_FLAVOR=s3 \
         -e AWS_BUCKET=acme-docker \
         -e STORAGE_PATH=/registry \
         -e AWS_KEY=AKIAHSHB43HS3J92MXZ \
         -e AWS_SECRET=xdDowwlK7TJajV1Y7EoOZrmuPEJlHYcNP2k4j49T \
         -e SEARCH_BACKEND=sqlalchemy \
         -p 5000:5000 \
         registry
```

此外，还可以指定本地路径（如 /home/user/registry-conf ）下的配置文件。

```
$ sudo docker run -d -p 5000:5000 -v /home/user/registry-conf:/registry-conf -
e DOCKER_REGISTRY_CONFIG=/registry-conf/config.yml registry
```

默认情况下，仓库会被创建在容器的 /var/lib/registry（v1 中是/tmp/registry）下。可以通过 -v 参数来将镜像文件存放在本地的指定路径。 例如下面的例子将上传的镜像放到 /opt/data/registry 目录。

```
$ sudo docker run -d -p 5000:5000 -v /opt/data/registry:/var/lib/registry
registry
```

本地安装

对于 Ubuntu 或 CentOS 等发行版，可以直接通过源安装。

- Ubuntu

```
$ sudo apt-get install -y build-essential python-dev libevent-dev python-pip
liblzma-dev
$ sudo pip install docker-registry
```

- CentOS

```
$ sudo yum install -y python-devel libevent-devel python-pip gcc xz-devel
$ sudo python-pip install docker-registry
```

也可以从 docker-registry 项目下载源码进行安装。

```
$ sudo apt-get install build-essential python-dev libevent-dev python-pip
libssl-dev liblzma-dev libffi-dev
$ git clone https://github.com/docker/distribution
$ cd distribution
$ sudo docker build .
```

启动运行registry的容器

```
$ sudo docker run -d -p 5000:5000 --restart=always --name registry ${IMAGE_ID}
```

###在私有仓库上传、下载、搜索镜像

创建好私有仓库之后，就可以使用 docker tag 来标记一个镜像，然后推送它到仓库，别的机器上就可以下载下来了。例如私有仓库地址为 10.42.123.13:5000。

先在本机查看已有的镜像。

```
$ sudo docker images
REPOSITORY                          TAG                 IMAGE ID
CREATED             SIZE
nginx                               v3                  706dd13cc995
About an hour ago   107MB
nginx                               v2                  c64380f0eb1a        14
hours ago        107MB
```

使用docker tag 将 706dd 这个镜像标记为 10.42.123.13:5000/test（格式为 docker tag IMAGE[:TAG] [REGISTRYHOST/][USERNAME/]NAME[:TAG]）。

```
$ sudo docker tag 706dd 10.42.123.13:5000/test
root ~ # docker images
REPOSITORY                          TAG                 IMAGE ID
CREATED             SIZE
10.42.123.13:5000/test              latest              706dd13cc995
About an hour ago   107MB
nginx                               v3                  706dd13cc995
About an hour ago   107MB
nginx                               v2                  c64380f0eb1a        14
hours ago        107MB
```

使用 docker push 上传标记的镜像。

```
$ sudo docker push 10.42.123.13:5000/test
The push refers to a repository [10.42.123.13:5000/test]
0a24c6c23fea: Pushed
af5bd3938f60: Pushed
29f11c413898: Pushed
eb78099fbf7f: Pushed
latest: digest:
sha256:dde58011e84b7746c8a309f9144102cf337e5252e9575b84610a1d7b9adc57e2 size:
1155
```

查看仓库中的镜像。

```
https://10.42.123.13:5000/v2/_catalog
```

这里可以看到 {"repositories":["test"]}，表明镜像已经被成功上传了。

现在可以到另外一台机器去下载这个镜像。

```
$ sudo docker pull 10.42.123.13:5000/test

$ sudo docker images
```

仓库配置文件

Docker 的 Registry 利用配置文件提供了一些仓库的模板（flavor），用户可以直接使用它们来进行开发或生产部署。

模板

在 config_sample.yml 文件中，可以看到一些现成的模板段：

- common：基础配置
- local：存储数据到本地文件系统
- s3：存储数据到 AWS S3 中
- dev：使用 local 模板的基本配置
- test：单元测试使用
- prod：生产环境配置（基本上跟s3配置类似）
- gcs：存储数据到 Google 的云存储
- swift：存储数据到 OpenStack Swift 服务
- glance：存储数据到 OpenStack Glance 服务，本地文件系统为后备

- glance-swift：存储数据到 OpenStack Glance 服务，Swift 为后备
- elliptics：存储数据到 Elliptics key/value 存储

用户也可以添加自定义的模版段。

默认情况下使用的模板是 dev，要使用某个模板作为默认值，可以添加 SETTINGS_FLAVOR 到环境变量中，例如

```
export SETTINGS_FLAVOR=dev
```

另外，配置文件中支持从环境变量中加载值，语法格式为 _env:VARIABLENAME[:DEFAULT]。

示例配置

```
common:
    loglevel: info
    search_backend: "_env:SEARCH_BACKEND:"
    sqlalchemy_index_database:
        "_env:SQLALCHEMY_INDEX_DATABASE:sqlite:////tmp/docker-registry.db"

prod:
    loglevel: warn
    storage: s3
    s3_access_key: _env:AWS_S3_ACCESS_KEY
    s3_secret_key: _env:AWS_S3_SECRET_KEY
    s3_bucket: _env:AWS_S3_BUCKET
    boto_bucket: _env:AWS_S3_BUCKET
    storage_path: /srv/docker
    smtp_host: localhost
    from_addr: docker@myself.com
    to_addr: my@myself.com

dev:
    loglevel: debug
    storage: local
    storage_path: /home/myself/docker

test:
    storage: local
    storage_path: /tmp/tmpdockertmp
```

不带安全认证的本地私有仓库

- 修改配置文件以绕过Https认证

  在"/etc/docker/"目录下，创建"daemon.json"文件。在文件中写入，保存退出后，重启docker。：

```
{ "insecure-registries":["10.42.205.169:5000"] }
```