Docker实践

极客学院出版

前言

Docker是一个开发的平台，用来为开发者和系统管理员构建、发布和运行分布式应用。本教程将通过实践实践让你体会Docker带来的方便与苦恼。

适用人群

本教程适合初接触 Docker 的开发人员。

学习前提

学习本教程前需对，云计算以及内核的基本知识有所了解。

更新日期	更新内容
2015-07-24	Docker 实践

目录

1 Docker 实践

一、Docker 是什么

docker 直译为码头工人。当它成为一种技术时，做的也是码头工人的事。官网是这样描述它的：“Docker是一个开发的平台，用来为开发者和系统管理员构建、发布和运行分布式应用。”也就是说，如果把你的应用比喻为货物，那么码头工人（Docker）就会迅速的用集装箱将它们装上船。快速、简单而有效率。`

它是用 Go 语言写的，是程序运行的“容器”（Linux containers），实现了应用级别的隔离（沙箱）。多个容器运行时互补影响，安全而稳定。

我喜欢它的原因就是快速部署，安全运行，不污染我的系统。

二、试用 Try it!

官方提供一个互动的小教程，让你很容易的了解 Docker 的基本用法，快去试试吧!

三、安装

官方直接支持 64 位 Linux 系统安装 Docker，但如果想在 32 位系统中运行，有人也进行了一些尝试，比如 32 Ubuntu 下，参考点击打开链接。

其他系统的安装请参考官网，下面说说我在 Ubuntu14.04 下的安装。

1.将镜像加入到程序源中：

```
~$ sudo sh -c "echo deb http://mirror.yandex.ru/mirrors/docker/ docker main > /etc/apt/sources.list.d/docker.list"
```

2.接着 update

```
$ sudo apt-get update
```

3.如果报错就 fix 掉它：

```
W: GPG error: http://mirror.yandex.ru docker Release: The following signatures couldn't be verified because the public
```

解决此错误：

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys D8576A8BA88D21E9
Executing: gpg --ignore-time-conflict --no-options --no-default-keyring --secret-keyring /tmp/tmp.RmJ1SUpsXX
gpg: requesting key A88D21E9 from hkp server keyserver.ubuntu.com
gpg: key A88D21E9: public key "Docker Release Tool (releasedocker) <docker@dotcloud.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

4.下载 docker：

```
$ sudo apt-get install lxc-docker
```

静静的等待它下载完成吧。

另外，这个命令也许会有帮助：

```
$ curl -sSL https://get.docker.com/ubuntu/ | sudo sh
```

四、初步使用

终端中输入 docker，打印出 docker 的命令列表：

```
Commands:
attach    Attach to a running container
build     Build an image from a Dockerfile
commit    Create a new image from a container's changes
cp        Copy files/folders from a container's filesystem to the host path
create    Create a new container
diff      Inspect changes on a container's filesystem
events    Get real time events from the server
exec      Run a command in a running container
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 log in to a Docker registry server
logout    Log out from a Docker registry server
logs      Fetch the logs of a container
port      Lookup the public-facing port that is NAT-ed to PRIVATE_PORT
pause     Pause all processes within a container
ps        List containers
pull      Pull an image or a repository from a Docker registry server
push      Push an image or a repository to a 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 on the Docker Hub
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
unpause   Unpause a paused container
version   Show the Docker version information
wait      Block until a container stops, then print its exit code
```

接下来就可以尝试使用这些命令了，不过在进行下一步之前，我们要先了解几个概念。

五、重要概念

1. image 镜像 镜像就是一个只读的模板。比如，一个镜像可以包含一个完整的 Ubuntu 系统，并且安装了 apache。 镜像可以用来创建 Docker 容器。 其他人制作好镜像，我们可以拿过来轻松的使用。这就是吸引我的特性。
2. Docker container 容器 Docker 用容器来运行应用。容器是从镜像创建出来的实例（好有面向对象的感觉，类和对象），它可以被启动、开始、停止和删除。
3. 仓库 这个好理解了，就是放镜像的文件的场所。比如最大的公开仓库是 Docker Hub。

六、几个简单的实践

1.search

搜索仓库中是否有 wordpress 这个博客镜像，如下：

```
$ docker search wordpress
NAME   DESCRIPTION STARS OFFICIAL   AUTOMATED
wordpress  The WordPress rich content management syst...  185  [OK]
```

2.下载这个镜像

```
$ docker pull wordpress
wordpress:latest: The image you are pulling has been verified
```

3.查看自己的镜像

```
$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         VIRTUAL SIZE
linc-wiki           latest          b5a1e34b01c2    27 hours ago    689.7 MB
ubuntu              latest          5ba9dab47459    5 days ago      188.3 MB
wordpress           latest          ecc04d6d638c    6 days ago      470 MB
```

4.简单的运行

运行 wordpress 要进行 mysql 的配置，为了演示 run，将 ubuntu 跑起来吧。

```
$ docker run -it ubuntu /bin/bash
root@46ff2a695ce1:/# echo "I am linc"
I am linc
```

至此，体验结束。后续会有更加精彩的实践等着我们，Docker，我们来了！

参考：

https://docs.docker.com/installation/ubuntulinux/

https://bitnami.com/stack/mediawiki

https://docs.docker.com/userguide/

https://dockerpool.com/static/books/docker_practice

http://zhumeng8337797.blog.163.com/blog/static/100768914201452411574380 6/

http://www.cnblogs.com/imoing/p/dockervolumes.html
https://github.com/docker/fig/issues/88

2

Docker 实践 2：用 Docker 搭建 hg-server

如果有人已经将 hg server 的 image 做好了，那么我还要自己作吗？答案是拿来用吧。

一、安装

用 hg 为关键词搜索，得出以下结果：

```
$ docker search hg
NAME                    DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
hgomez/gatling                                                          1                    [OK]
v7soft/hgdns                                                            0                    [OK]
hg8496/gridvis-service                                                  0                    [OK]
hgomez/di-centos6-myjenkins-lts                                         0                    [OK]
jrandall/hgi-project                                                    0                    [OK]
hgomez/di-centos6-myartifactory                                         0                    [OK]
hgomez/di-centos6-myjenkins                                             0                    [OK]
hgomez/di-centos6-mynexus                                               0                    [OK]
hgomez/di-centos6-myarchiva                                             0                    [OK]
hg8496/piwigo                                                           0                    [OK]
hg8496/apache                                                           0                    [OK]
hgomez/di-centos6-mygitblit                                             0                    [OK]
hgomez/di-centos6-mygitbucket                                           0                    [OK]
jyotisingh/ubuntu-hg                                                    0
hg8496/dokuwiki                                                         0                    [OK]
hg8496/owncloud                                                         0                    [OK]
misshie/ucsc-blat-hg19                                                  0                    [OK]
ussie/hg-exec           adds mercurial to ubuntu:14.04.                 0
misshie/ucsc-blat-hg38                                                  0                    [OK]
hg8496/gridvis-pc                                                       0                    [OK]
        Test. Automated builds for this repo are b...                   0                    [OK]
hg8496/rsync                                                            0                    [OK]
secondbit/hgbundler                                                     0
uotbw/hgamer3d          Docker image for hgamer3d, see www.hgamer3...   0
hgomez/di-centos6-base                                                  0                    [OK]
```

hgweb 貌似不错的选择，在 github 上的主页是 https://github.com/amclain/docker-hgweb 。将其 pull 下来，在漫长的等待中我也在思考着如何启动它。

主页上提供了它的 Dockerfile，通过它我们就可以了解这个 image 是如果构造的。先来说说什么是 Dockerfile。

二、Dockerfile

它是用户创建自定义镜像的文件。它通常分为四部分：基础镜像信息，维护者信息，镜像操作指令和容器启动时的指令。

```
#基础系统信息，基于ubuntu 14.04构建的
FROM ubuntu:14.04
MAINTAINER Alex McLain <alex@alexmclain.com>
RUN apt-get -qq update
#安装apache、hg、php5
RUN apt-get -y install apache2 apache2-utils curl mercurial php5 php5-cli php5-mcrypt
# TODO: Remove
#是的，vim确实很大，不安装为好
RUN apt-get -y install vim
RUN echo "colorscheme delek" > ~/.vimrc
# Configure hgweb
ADD hg/add.php /etc/default/hgweb/hg/
ADD hg/hgweb.config /etc/default/hgweb/hg/
ADD hg/hgweb.cgi /etc/default/hgweb/hg/
ADD hg/hgusers /etc/default/hgweb/hg/
# Configure Apache
ADD apache/hg.conf /etc/default/hgweb/apache/
RUN rm /etc/apache2/sites-enabled/*
RUN a2enmod rewrite && a2enmod cgi
ADD load-default-scripts /bin/
RUN chmod u+x /bin/load-default-scripts
#创建一个挂载点，本机或其他容器可以将其挂载。启动时用-v参数进行挂载
VOLUME /var/hg
VOLUME /etc/apache2/sites-available
#暴露的端口号，启动时要通过-p参数指定
EXPOSE 80
#启动时执行的命令
CMD load-default-scripts && service apache2 start && /bin/bash
```

三、启动

有了上述的背景，我们知道启动时要做两件事：指定端口号、挂载本地目录。

比如还是使用端口号 80，那么只需用 -p 80:80即可。

比如本机目录 hg-repos 用来做 hg repo 的放置目录，只需 -v /home/linc/hg-repos:/var/hg/repos 挂载即可。

另外，我们还要将其启动在后台（Daemonized），加上-d 参数。

完整启动命令如下：

```
docker run -idt -p 80:80 -v /home/linc/hg-repos:/var/hg/repos amclain/hgweb
```

四、与后台容器交互

1.attach 方法

docker 自带 attach 命令，但此命令的不方便之处在于，多个窗口（同时 attach 此容器）会同步显示操作，并且当一个窗口 exit 时，所有窗口都会退出，后台运行的容器也停止了。

```
$ docker ps
CONTAINER ID        IMAGE                     COMMAND             CREATED             STATUS              PORTS               N
b22cc1880b7a        amclain/hgweb:latest      "/bin/sh -c 'load-de   3 hours ago         Up 3 hours          0.0.0.0:80->80/t

$ docker attach b22cc1880b7a
root@b22cc1880b7a:/#
```

2.nsenter

此工具需要从源码安装：

```
$ cd /tmp; curl https://www.kernel.org/pub/linux/utils/util-linux/v2.24/util-linux-2.24.tar.gz | tar -zxf-; cd util-linux-2.24;
$ ./configure --without-ncurses
$ make nsenter && sudo cp nsenter /usr/local/bin
```

直接用 nsenter 命令交互很繁琐，然后有人写了配置文件放到 bashrc 中，就可以方便的使用了。

```
#docker
export DOCKER_HOST=tcp://localhost:4243
alias docker-pid="sudo docker inspect --format '{{.State.Pid}}'"
alias docker-ip="sudo docker inspect --format '{{ .NetworkSettings.IPAddress }}'"

#the implementation refs from https://github.com/jpetazzo/nsenter/blob/master/docker-enter
function docker-enter() {
if [ -e $(dirname "$0")/nsenter ]; then
# with boot2docker, nsenter is not in the PATH but it is in the same folder
NSENTER=$(dirname "$0")/nsenter
else
NSENTER=nsenter
fi
[ -z "$NSENTER" ] && echo "WARN Cannot find nsenter" && return

if [ -z "$1" ]; then
```

```
echo "Usage: `basename "$0"` CONTAINER [COMMAND [ARG]...]"
echo ""
echo "Enters the Docker CONTAINER and executes the specified COMMAND."
echo "If COMMAND is not specified, runs an interactive shell in CONTAINER."
else
PID=$(sudo docker inspect --format "{{.State.Pid}}" "$1")
if [ -z "$PID" ]; then
echo "WARN Cannot find the given container"
return
fi
shift

OPTS="--target $PID --mount --uts --ipc --net --pid"

if [ -z "$1" ]; then
# No command given.
# Use su to clear all host environment variables except for TERM,
# initialize the environment variables HOME, SHELL, USER, LOGNAME, PATH,
# and start a login shell.
#sudo $NSENTER "$OPTS" su - root
sudo $NSENTER --target $PID --mount --uts --ipc --net --pid su - root
else
# Use env to clear all host environment variables.
sudo $NSENTER --target $PID --mount --uts --ipc --net --pid env -i $@
fi
fi
}
```

其中有两个 alias 和一个 function，使用 docker-enter 会很容易于容器交互并没有 attach 中的副作用。如下：

```
$ docker ps
CONTAINER ID        IMAGE                       COMMAND             CREATED             STATUS              PORTS               N
beb178cd9335        amclain/hgweb:latest        "/bin/sh -c 'load-de   11 seconds ago      Up 10 seconds       0.0.0.0:80->

$ docker-enter beb178cd9335
root@beb178cd9335:~# ls
root@beb178cd9335:~# pwd
/root
```

五、快速启动 hg-server

咱也写个 alias 放子 bashrc 中，如下：

```
alias docker-load-hg-server="sudo docker run -idt -p 80:80 -v /home/linc/hg-repos:/var/hg/repos amclain/hgweb"
```

启动它：

```
$ docker-load-hg-server
[sudo] password for linc:
beb178cd933502970fd12d9a4babecef5475a52d85a207066c665b4a620c5a62
```

改进

对于文件的挂载，其实直接挂镜像的/var/hg更好，这样里面的几个配置文件如 hgusers hgweb.cgi hgweb.config，我们可以直接进行配置。

```
sudo docker run -idt -p 80:80 -v /home/linc/hg-repos:/var/hg amclain/hgweb
```

3

Docker 实践 3：fig 搭建 mediawiki

fig，无花果。fig 项目源自 docker，可以认为是快速搭建基于 Docker 的隔离开发环境的工具。

一、安装 fig

```
$ mkdir docker; cd docker

$ curl -L https://github.com/docker/fig/releases/download/1.0.1/fig-`uname -s`-`uname -m` > fig

$ sudo chmod +x fig; sudo mv fig /usr/local/bin/
```

二、搭建 mediawiki

使用个人构建的镜像，github 地址：https://github.com/bopjiang/wikimedia-docker

在 docker 目录下执行如下命令：

```
$ git clone https://github.com/bopjiang/wikimedia-docker.git
$ cd wikimedia-docker
$ fig up -d
```

此时有两个容器启动：

```
r$ docker ps
CONTAINER ID        IMAGE                          COMMAND                CREATED             STATUS              PORTS                    N
21182a060c17        nickstenning/mediawiki:latest   "/usr/bin/mediawiki-   7 hours ago         Up 7 hours          0.0.0.0:8880-
728ec09c3552        mysql:5.7.5                    "/entrypoint.sh mysq   7 hours ago         Up 7 hours          3306/tcp                 wik
```

三、fig.yml

fig.yml 用来配置镜像构建的具体内容，此 wiki 的 fig.yml 在 wikimedia-docker 目录下，内容如下：

```
wiki2:
    image: 'nickstenning/mediawiki'
    ports:
        - "8880:80"
    links:
        - db:database
    volumes:
        - /data/wiki2:/data

db:
    image: "mysql:5.7.5"
    expose:
        - "3306"
    environment:
        - MYSQL_ROOT_PASSWORD=defaultpass
```

image：用来指定镜像，如果本地没有，fig 将会尝试去远程 pull 这个镜像。

ports：暴露的端口.

links：在其他服务中连接容器。

volumes: 卷挂载路径，容器中的/data/目录挂载到主机的/data/wiki2下。在 wiki 配置完毕后，将 LocalSettings.php 文件放置在主机的/data/wiki2 目录下。

expose：也是暴露端口，与 ports 的区别是不发布到宿主机的端口，只被连接的服务访问。

environment：设置环境变量。

四、wiki 的配置

浏览器中输入 localhost:8880,首次启动会让进入配置界面。完成后生成 LocalSettings.php 文件。也可以直接在这个配置文件中作配置。

生成的 LocalSettings.php 文件要拷到/data/wiki2目录下（配置文件中定义的卷挂载路径），并增加其 r 属性就可以了。

还记得在 yml 配置文件中数据库主机名是什么吧？database，对了，那么在配置中也要这样填写，如图：

图片 3.1 picture3.1

五、wiki 的使用技巧

1.左侧导航栏的配置

以管理员身份登录，在搜索栏中输入 MediaWiki:sidebar

进入配置界面后就可以编辑了。比如：

```
<pre>
navigation
   http://192.168.0.111:8880/index.php?title=Category:XXX|XXX
   mainpage|mainpage-description
    portal-url|portal
</pre>
```

3.分类

文章的末尾加入"category"标签即可将此文章放到了 xxx 分类中，一篇文章可以加入多个分类。

比如：[[category:XXX]]

4.新文章

在 Search 中输入你的文章名称即可 Edit。

5.换行

用 br 标签可以换行。

空一行也会有换行效果。

6.pre 标签包围源代码

例如：

```
<pre>
private int mSize;
</pre>
```

六、保存容器和导入

```
sudo docker commit 9ab6e234c9ba linc-wiki

sudo docker images REPOSITORY            TAG             IMAGE ID        CREATED         VIRTUAL SIZE linc-wiki

sudo docker export 9ab6e234c9ba > /home/linc/docker/images-bk/linc-wiki-export.tar
sudo docker save linc-wiki > ../images-bk/linc-wiki-save.tar

$ du -sh *
495M    linc-wiki-export.tar
672M    linc-wiki-save.tar

sudo cat /home/linc/docker/images-bk/linc-wiki-export.tar | sudo docker import - docker_hgweb
sudo docker load --input ../images-bk/linc-wiki-save.tar
```

附录：

1.fig 使用报错及解决

fig running error:

```
$ fig up
Couldn't connect to Docker daemon at http:/ - is it running?

If it's at a non-standard location, specify the URL with the DOCKER_HOST environment variable.
```

fix it:

```
1) Change the DOCKER_OPTS in /etc/default/docker to:
DOCKER_OPTS="-H tcp://127.0.0.1:4243 -H unix:///var/run/docker.sock"

2) Restart docker
sudo restart docker

3) Make sure that docker is running on localhost:4243
$ netstat -ant  |grep 4243
tcp0  0 127.0.0.1:4243  0.0.0.0:*   LISTEN

4) Set DOCKER_HOST (.bashrc)
```

```
export DOCKER_HOST=tcp://localhost:4243

$ echo $DOCKER_HOST
tcp://localhost:4243
```

参考：

dockerpool.com/static/books/docker_practice/fig/yml_ref.html

4

Docker 实践 4：搭建 wordpress

在系列的第一篇文章《 Docker 实践 》中已经 search 到并 pull 了官方的 wordpress 镜像，接下来我们还要 search 一个官方的 mysql 将二者结合，搭建一个可用的 wordpress 站点。

首先，搞定 mysql

1.search

```
$ docker search mysql
NAME DESCRIPTION STARS OFFICIAL   AUTOMATED
mysqlMySQL is a widely used, open-source relati...   456   [OK]
```

2.pull

```
$ docker pull mysql
```

其次，考虑二者的联合

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
wordpress           latest              ecc04d6d638c        2 weeks ago         470 MB
mysql               latest              aca96d9e6b5c        2 weeks ago         282.7 MB
```

wordpress 启动命令是这样的：

```
$ sudo docker run --name some-wordpress --link some-mysql:mysql -d wordpress
```

```
启动 WordPress 容器时可以指定的一些环境参数包括

-e WORDPRESS_DB_USER=... 缺省为 “root”
-e WORDPRESS_DB_PASSWORD=... 缺省为连接 mysql 容器的环境变量 MYSQL_ROOT_PASSWORD 的值
-e WORDPRESS_DB_NAME=... 缺省为 “wordpress”
-e WORDPRESS_AUTH_KEY=..., -e WORDPRESS_SECURE_AUTH_KEY=..., -e WORDPRESS_LOGGED_IN_KE
```

针对 wordpress 的启动命令，我们需要这样针对：

1. 给 wordpress 可以起个名字，这个好办
2. --link 参数，这需要我们先启动 mysql，然后将其名字链接上
3. 端口 -p 参数，默认是 80 端口，但是被我占用了，这里我们映射到 8080

启动的 mysql 的命令：

```
$ docker run --name mysql_wordpress -e MYSQL_ROOT_PASSWORD=wordpress  -d  mysql
```

mysql 的密码，姑且这样暴露着吧。

对应 mysql，wordpress 的启动命令如下：

```
$ docker run --name docker_wordpress --link mysql_wordpress:mysql -p 8080:80 -d wordpress
```

接下来就可以在浏览器中输入 http://localhost:8080 进行 wordpress 的配置了。

第三，用 fig 来配置

实践证明，用 fig 配置是最好的途径。在上面的基础上，我们只需在自己的 docker 目录下新建目录如 wordpress-docker，再建 fig 配置文件 fig.yml 如下：Enjoy!

```
wordpress:
    image: "wordpress:latest"
    ports:
        - "8080:80"
    links:
        - db:mysql

db:
    image: "mysql:latest"
    expose:
        - "3306"
    environment:
        - MYSQL_ROOT_PASSWORD=wordpress
```

每次启动只需执行本目录下的 fig up -d 就可以了!

参考：
https://github.com/docker-library/wordpress/blob/aee00669e7c43f435f021cb02871bffd63d5677a/Dockerfile

如果想用 fig 搭建 wordpress，个人感觉更方便一些，参考如下网址：
http://dockerpool.com/static/books/docker_practice/fig/wordpress.html

5

Docker 实践 5：搭建 redmine

Redmine 是一个开源的项目管理系统，它有如下优势让我选择它作为我的项目管理工具。

1. 支持多项目管理
2. 灵活的角色管理
3. 灵活的 issue/bug 跟踪管理
4. 支持甘特图和日历
5. 支持新闻、文档和文件管理，邮件通知等功能
6. 每个项目有自己的 wiki 和论坛，这一点非常棒
7. 与 SCM 系统集成，支持 SVN, CVS, Git, Mercurial, Bazaar and Darcs 等源代码管理工具，这一点同样非常棒

有了 Redmine，让项目经理不用愁管理项目了。

同样，看看官方是否出 docker 镜像或者其他人作好镜像了，我直接用就好了。

```
$ docker search redmine
NAME                    DESCRIPTION                     STARS    OFFICIAL   AUTOMATED
sameersbn/redmine                                       72                  [OK]
```

这与在 docker hub 上搜索是一样的，虽然没有官方的镜像，那我们就选择星星数量多的镜像，sameersbn/redmine 就成了我的选择。

用 fig 直接快速安装，在自己的 docker 目录下新建 redmine 目录，在里面执行：

```
~/docker/redmine$ wget https://raw.githubusercontent.com/sameersbn/docker-redmine/master/fig.yml
```

下载的 fig.yml 内容如下：

```
postgresql:
  image: sameersbn/postgresql:9.1-1
  environment:
    - DB_USER=redmine
    - DB_PASS=phatiphohsukeuwo
    - DB_NAME=redmine_production
redmine:
  image: sameersbn/redmine:2.6.1
  links:
    - postgresql:postgresql
  environment:
```

```
    - DB_USER=redmine
    - DB_PASS=phatiphohsukeuwo
    - DB_NAME=redmine_production
  ports:
    - "10080:80"
```

直接快速启动就可以了。

```
~/docker/redmine$ fig up -d
```

```
$ docker ps
CONTAINER ID        IMAGE                          COMMAND                CREATED             STATUS              PORTS
5d5d5a983298        sameersbn/redmine:2.6.1        "/app/init app:start   51 minutes ago      Up 51 minutes       443/tcp, 0.0
c78a212c1503        sameersbn/postgresql:9.1-1     "/start"               About an hour ago   Up About an hour    5432/tcp
```

浏览器中输入 http://localhost:10080,

管理员帐号是 admin，密码 admin。

参考：

https://registry.hub.docker.com/u/sameersbn/redmine/

Docker 实践 6：Cannot connect to the Docker daemon.

正在免费适用着 Aliyun 主机，当然要用 docker 来部署我的服务器啦。但是今天碰到了题目的问题，细节如下：

```
# docker info
FATA[0000] Cannot connect to the Docker daemon. Is 'docker -d' running on this host?
# docker -d
INFO[0000] +job serveapi(unix:///var/run/docker.sock)
FATA[0000] pid file found, ensure docker is not running or delete /var/run/docker.pid
```

首先要查看 docker daemon 是否在运行。

```
# ps aux | grep docker
root   691  0.0  1.6 506388 17068 ?Ssl  Mar07   0:45 /usr/bin/docker -d
root  1330  0.0  0.8 124088  8392 ?Sl   Mar07   0:01 docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 80 -container
root  9607  0.0  0.0  11720   896 pts/2S+   16:50   0:00 grep --color=auto docker
```

这样看来，docker deamon 正在运行，但是报此错误实属不应该。那么将其停止，再启动。

```
# service docker stop
# ps aux | grep docker
root  9624  0.0  0.0  11716   636 pts/2S+   16:52   0:00 grep --color=auto docker
# docker -d
INFO[0000] +job serveapi(unix:///var/run/docker.sock)
INFO[0000] Listening for HTTP on unix (/var/run/docker.sock)
INFO[0000] +job init_networkdriver()
INFO[0000] -job init_networkdriver() = OK (0)
INFO[0000] WARNING: Your kernel does not support cgroup swap limit.
INFO[0000] Loading containers: start.
.....
INFO[0001] Loading containers: done.
INFO[0001] docker daemon: 1.5.0 a8a31ef; execdriver: native-0.2; graphdriver: aufs
INFO[0001] +job acceptconnections()
INFO[0001] -job acceptconnections() = OK (0)
```

好吧，最有趣的事情是，之前我在 root下一切运行良好，但是现在就像见鬼一样遇到上面的问题。上述的工作完成后，仍有问题。最后我使用 sudo 竟然解决了问题。如下：

```
# sudo docker info
Containers: 5
Images: 32
Storage Driver: aufs
 Root Dir: /var/lib/docker/aufs
 Backing Filesystem: extfs
```

```
 Dirs: 42
Execution Driver: native-0.2
Kernel Version: 3.13.0-32-generic
Operating System: Ubuntu 14.04.1 LTS
```

7

Docker 实践 7：容器与主机拷贝数据

在 Docker 实践 2 中使用-v 参数将主机与容器中相关目录联系在一起（挂载），所以我们可以用这个通道将想要互相拷贝的数据放入其中，这样就可以用 cp 命令来复制文件了。

除了这个办法，我们还可以分别用不同的命令来拷贝数据。

从容器中像主机拷贝数据

docker 提供了 cp 命令，用法如下：

```
# docker ps
CONTAINER IDIMAGE COMMANDCREATED STATUS  PORTS NAMES
a77a72ac178ctutum/apache-php:latest   "/run.sh"  21 hours agoUp 21 hours 0.0.0.0:8080->80/tcp  phpapache_phpap
# docker-enter a77a72ac178c
root@a77a72ac178c:~# ls /var/www/html
index.php  logo.png
root@a77a72ac178c:~# exit
logout

# docker cp a77a72ac178c:/var/www/html /var/www/
# ls /var/www/
app  download  index.html
# ls /var/www/app/
index.php  logo.png
```

从主机向容器中拷贝数据

这里要使用一个 docker 提供的神奇通道来完成主机向容器的数据传输。

首先要用 docker inspect 方法获得容器的完整 id，

```
inspect   Return low-level information on a container
```

然后用/var/lib/docker/aufs/mnt/通道来完成拷贝。

举例如下：

```
# docker inspect -f '{{.Id}}' a77a72ac178c
a77a72ac178c1e35708d2af446197c10239b0b1bd8932104578e334b83eb93a2
# cp docker/docker-start.sh /var/lib/docker/aufs/mnt/a77a72ac178c1e35708d2af446197c10239b0b1bd8932104578e33
# docker-enter a77a72ac178c
# pwd
/root
# ls
docker-start.sh
```

8

Docker 实践 8：Compose

今天要在我的本子上搭建一个 mediawiki 环境，之前的经验，用 fig 去配置是最简单的了。可是下载 fig 失败，去官网一看才知道，fig 已经被 compose 工具取代了。原文是这样说的：

> Fig has been replaced by Docker Compose, and is now deprecated. The new documentation is on the Docker website.

既然如此，就去官网看看 compose 到底为何物。

compose 是用来在 docker 中定义和运行复杂应用的小工具，比如在一个文件中定义多个容器，只用一行命令就可以让一切就绪并运行。它的功能与我们所熟知的 fig 相似，换句话说，compose 是 fig 的替代产品，fig 就这样退出 docker 的历史舞台了。

然而在 github 上的 compose 有这样的说法：

> Fig has been renamed to Docker Compose, or just Compose for short. This has several implications for you:
>
> The command you type is now docker-compose, not fig. You should rename your fig.yml to docker-compose.yml.

看来 fig 是被重命名成 compose 了，配置文件变成了 docker-compose.yml，其他都几乎一样。不但 fig 不能下载了，原来有 fig 工具的环境用 fig 去搭建 mediawiki 都不可用了，报错如下：

```
fig up -d
Creating hgserver_mediawiki_1...
Pulling image amclain/hgweb...
Traceback (most recent call last):
  File "<string>", line 3, in <module>
  File "/code/build/fig/out00-PYZ.pyz/fig.cli.main", line 31, in main
  File "/code/build/fig/out00-PYZ.pyz/fig.cli.docopt_command", line 21, in sys_dispatch
  File "/code/build/fig/out00-PYZ.pyz/fig.cli.command", line 28, in dispatch
  File "/code/build/fig/out00-PYZ.pyz/fig.cli.docopt_command", line 24, in dispatch
...
fig.progress_stream.StreamOutputError: Get https://index.docker.io/v1/repositories/amclain/hgweb/images: dial tcp: lo
```

如此看来，使用 compose 是必须的了。

下面说说 compose 的用法。

1.安装 compose

OS X 和 64 位的 Linux 用如下命令安装。

```
# curl -L https://github.com/docker/compose/releases/download/1.1.0/docker-compose-`uname -s`-`uname -m` > /u

#chmod +x /usr/local/bin/docker-compose
```

其他平台可以像 python 包一样安装：

```
$ sudo pip install -U docker-compose
```

2.命令简介

```
$ docker-compose
Fast, isolated development environments using Docker.

Usage:
  docker-compose [options] [COMMAND] [ARGS...]
  docker-compose -h|--help

Options:
  --verbose               Show more output
  --version               Print version and exit
  -f, --file FILE         Specify an alternate compose file (default: docker-compose.yml)
  -p, --project-name NAME   Specify an alternate project name (default: directory name)

Commands:
  build     Build or rebuild services
  help      Get help on a command
  kill      Kill containers
  logs      View output from containers
  port      Print the public port for a port binding
  ps        List containers
  pull      Pulls service images
  rm        Remove stopped containers
  run       Run a one-off command
  scale     Set number of containers for a service
  start     Start services
  stop      Stop services
  restart   Restart services
  up        Create and start containers
```

3.compose 编写 mediawiki 的 docker-compose.yml

首先编写 compose 的配置文件，语法与 fig 类似，文件名为 docker-compose.yml，内容如下：

```
wiki2:
    image: 'nickstenning/mediawiki'
    ports:
        - "8880:80"
    links:
        - db:database
    volumes:
        - /data/wiki2:/data

db:
    image: "mysql"
    expose:
        - "3306"
    environment:
        - MYSQL_ROOT_PASSWORD=defaultpass
```

4.创建并启动 mediawiki

```
$ docker-compose up -d
```

9

Docker 实践 9：备份方案

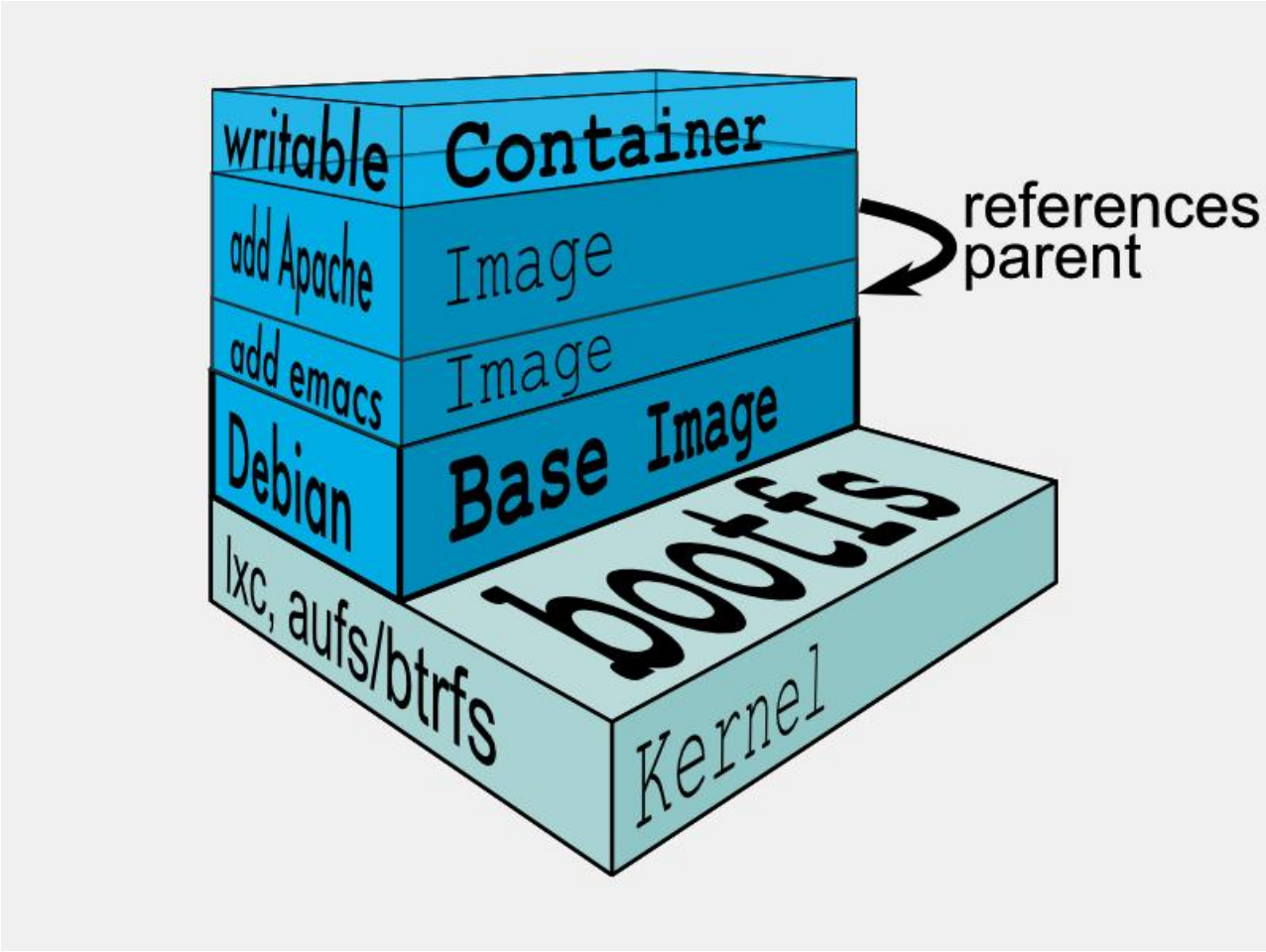
1 两个文件系统

先提一下两个重要的文件系统概念，一个是 aufs，一个是 vfs.

aufs 是一个类似于 Unionfs 的可堆叠联合文件系统。它将多个目录整合成单一的目录。ubuntu 对其有良好的支持，因此 docker 的镜像就存储在 aufs 文件系统下。

vfs 是 linux 的内核中一个重要概念，这个虚拟文件系统可以让 open()、read()、write()等系统调用不用关心底层的存储介质和文件系统类型就可以工作的粘合层。

2 docker 镜像与容器的存储



图片 9.1 picture9.1

docker 的层次结构如上图。

docker 安装在/var/lib/docker 目录，来这里访问需要 root 权限。

```
/var/lib/docker# du -sh *
 1.8G    aufs
80K containers
36K execdriver
452K    graph
15M init
 8.0K    linkgraph.db
 4.0K    repositories-aufs
 240M    tmp
8.0K    trust
```

```
203M    vfs
28K volumes
```

从文件夹大小就可以看出 aufs 和 vfs 是实际存东西的，在最开始的时候，我的只把重点放在了 aufs 上。

当前有两个 docker 镜像，使用 docker-compose 部署的 mediawiki。

```
# docker images
REPOSITORY              TAG             IMAGE ID        CREATED         VIRTUAL SIZE
mysql                   5.7.5           c171a2256c3b    3 weeks ago     322.8 MB
nickstenning/mediawiki  latest          d32d732341b0    16 months ago   619.5 MB
```

与之对应的 repositories-aufs 如下：

```
/var/lib/docker# cat repositories-aufs | python -mjson.tool
{
    "Repositories": {
        "mysql": {
            "5.7.5": "c171a2256c3b9d13c77b2b3773eaaad95a9b4a8735613ce46722f79fa29fa366"
        },
        "nickstenning/mediawiki": {
            "latest": "d32d732341b07cfc87006d09ea75514d8c10774dfbc83c74034ac28b1491557b"
        }
    }
}
```

趁着 tree 参数可用，我们来看看 images 的层级结构：

```
# docker images -tree
Warning: '-tree' is deprecated, it will be removed soon. See usage.
├─511136ea3c5a Virtual Size: 0 B
│ └─36669626e49c Virtual Size: 84.98 MB
│   └─d5570ef1464a Virtual Size: 84.98 MB
│     └─170ff2cfa64a Virtual Size: 85.3 MB
│       └─ef7c16d26957 Virtual Size: 116.7 MB
│         └─3749ee37c2a6 Virtual Size: 116.8 MB
│           └─5431d6b9bc80 Virtual Size: 116.8 MB
│             └─0dc7d8240120 Virtual Size: 116.8 MB
│               └─a37379b8d1ce Virtual Size: 116.8 MB
│                 └─cad6baf3ee5b Virtual Size: 322.8 MB
│                   └─facdf8305638 Virtual Size: 322.8 MB
│                     └─a10cdfaf13cf Virtual Size: 322.8 MB
│                       └─444b88bd8367 Virtual Size: 322.8 MB
```

```
|                   └─83dfd5776ff8 Virtual Size: 322.8 MB
|                     └─7d0b9c0b29d2 Virtual Size: 322.8 MB
|                       └─c171a2256c3b Virtual Size: 322.8 MB Tags: mysql:5.7.5
└─8dbd9e392a96 Virtual Size: 128 MB
  └─be647841828f Virtual Size: 128 MB
    └─7f06ad8f23b1 Virtual Size: 272.5 MB
      └─9fc1b767f7ff Virtual Size: 383.1 MB
        └─46af893ffb5f Virtual Size: 439.2 MB
          └─00a79ad7ea4a Virtual Size: 439.2 MB
            └─69f7b013792e Virtual Size: 439.2 MB
              └─854a5bd5be72 Virtual Size: 439.2 MB
                └─c219ca4d0d53 Virtual Size: 439.2 MB
                  └─42c69a7e8ad3 Virtual Size: 439.2 MB
                    └─09652ed5f7ef Virtual Size: 459.6 MB
                      └─814a97e8dc1c Virtual Size: 539.6 MB
                        └─a7bc400b42c2 Virtual Size: 539.6 MB
                          └─68013db1eb6d Virtual Size: 619.5 MB
                            └─ca839611a34f Virtual Size: 619.5 MB
                              └─63e9d31943ea Virtual Size: 619.5 MB
                                └─1b95e1c4efd9 Virtual Size: 619.5 MB
                                  └─9df1e07fdf78 Virtual Size: 619.5 MB
                                    └─3e8e8966dcfb Virtual Size: 619.5 MB
                                      └─3e0c3340e16e Virtual Size: 619.5 MB
                                        └─d32d732341b0 Virtual Size: 619.5 MB Tags: nickstenning/mediawiki:latest
```

再去看看 graph 目录：

```
/var/lib/docker/graph# ls
00a79ad7ea4a77bac24386226563b86ee92db49073e6417ce10dbe175777b7ff  814a97e8dc1c5b86b3a4a21e2c64abac
09652ed5f7efeb035c7f29e7fe16b600ff4066362ea95f678974d7940f89f021  83dfd5776ff844e112fa504701370e7a0e20e
0dc7d82401208db703d519db581d7dddb39a090e2bf0dd6b5a64dcf9a743e6aa  854a5bd5be7280e1a53d451e3806aaf2
170ff2cfa64adbd757c0dad0c1cb4bbb61e11301a16cd8f29b40cb3217ec7254  8dbd9e392a964056420e5d58ca5cc376ef
1b95e1c4efd993116df499fa9951b865ec99e8a455427cffc9c61364bc3c5295  9df1e07fdf780918f8fafaba0f5470d1c63572
36669626e49c623f3b2aea41052bbc9de0506807d92137bf76c0fd3264eaff34  9fc1b767f7ff6d935efd6e897882d202fcbc
3749ee37c2a6df4df9f720c7c07ecc5af695a2a5853bac1f8ba6450ddc310343  a10cdfaf13cf8dbaa07798f29265ec37a5be
3e0c3340e16e991af227912b27441b392762476f15d6749c3c1333fab63c5927  a37379b8d1ce392404d50d97a02afb92b
3e8e8966dcfba21708e772ec165edcb617061028572e279decf7b374f4f3a866  a7bc400b42c28673642376884b7a8a93c
42c69a7e8ad3ba37bfddeaf52ea54381d487311ba5c75d6a91811abeaad8438b  be647841828f846812df2968e24aa9db4
444b88bd8367b8fc2d023b5ceb77cec3784963bbfc03d1d30f96b97a22cfbc44  c171a2256c3b9d13c77b2b3773eaaad95
46af893ffb5fe47b7de23a3f2f8972dbf51e78a0327ca46b7848c1cdfd48f97d  c219ca4d0d536c79955628e17aa8d6e6b6a
511136ea3c5a64f264b78b5433614aec563103b4d4702f3ba7d4d2698e22c158  ca839611a34f4eb30e582694d9a1cdabf
5431d6b9bc80429667488287e8189cc4d4b14791e43e45b374e132e340fb15db  cad6baf3ee5b2798947cdcfbdd3bf4c38
63e9d31943ea386e435763524d9ea3a8cb253747113abbb53191121b815572da  d32d732341b07cfc87006d09ea75514d
68013db1eb6d12803db7cbb52e95e6f29c8141c095b8cb50d245925d7f70d774  d5570ef1464a43fe282dd2705b38a2d73
69f7b013792e1f9c63176a7e348cb38277182f8bcf3d1d6d2c26ff9c4885b4ec  ef7c16d26957dc5dea9ad23433c7dd6400a
```

```
7d0b9c0b29d271a8be6ca9252da15ae4f01dd386ac2b452563232aaaddd5eaf9  facdf830563864336b802331675773155
7f06ad8f23b10227423823e3acc7e368c012f41a30660c3fbad661a9a4f8fe67  _tmp
```

长 id 与 tree 的短 id 是对应的，这样就明晰了。在上文中给 container 传文件用到了神奇通道，其实就是 aufs 中的 mnt。

备份方案

第一种情况，挂载 host 目录的，比如之前的 hg-server。备份就很简单，直接把那些 hg 源码目录在 host 做备份即可。

第二种情况，需要在 container 中安装新软件。用 export 或者 save 命令吧。

```
sudo docker commit 9ab6e234c9ba linc-wiki

sudo docker images REPOSITORY   TAG IMAGE IDCREATED VIRTUAL SIZE linc-wikilatest  b5a1e34b01c214 second

sudo docker export 9ab6e234c9ba > /home/linc/docker/images-bk/linc-wiki-export.tar
sudo docker save linc-wiki > ../images-bk/linc-wiki-save.tar

$ du -sh *
495M    linc-wiki-export.tar
672M    linc-wiki-save.tar

sudo cat /home/linc/docker/images-bk/linc-wiki-export.tar | sudo docker import - docker_hgweb
sudo docker load --input ../images-bk/linc-wiki-save.tar
```

第三种情况，mysql 中的数据哪里去啦?

这个问题困扰我几天了。就拿部署的 mediawiki 说吧，没有将数据文件链接到 host 中，而是放到了/var/lib/mysql中，如下:

```
 /var/lib/mysql# ls
auto.cnf  ib_logfile0  ib_logfile1  ibdata1  ibtmp1  my_wiki  mysql  performance_schema
```

我嘗試着新设置 volume 不过没有成功，也做了其他努力，都失败了。最后一次大搜索中，发现这些数据文件被放在了 vfs 中。

```
/var/lib/docker/vfs# tree -L 3
.
└── dir
    └── cb4012594631102fc8a69aa6cb4a9aa2dd2be3d39d2564c94ae91ac5e3eb4aec
        ├── auto.cnf
        ├── ibdata1
        ├── ib_logfile0
        ├── ib_logfile1
```

```
        ├── ibtmp1
        ├── mysql
        ├── my_wiki
        └── performance_schema

5 directories, 5 files
```

尝试了将其放入我的 ubuntu server 虚拟机中的 docker 相同目录下，mediawiki 用 host 中 save 后的镜像，我的 mediawiki 在 ubuntu server 虚拟机中完美复活！

也就是说，我只需要备份 host 系统中 docker 下的 vfs 的数据就可以了。问题解决，可以肆意在 wiki 中添加重要内容而不担心被毁掉了。

参考：

http://www.programfish.com/blog/?p=9

http://blog.csdn.net/junjun16818/article/details/38423391

gitlab 的维护：http://www.tuicool.com/articles/bYbi2mJ

极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台

更多信息请访问

http://wiki.jikexueyuan.com/project/docker-practice/