

1、为什么要用docker

资源 环境独立，运维部署方便。易于扩展。

关于docker的知识：

镜像 Images

容器 container

镜像和容器 的关系 类似于 类和 对象， 容器是镜像的一个运行实例。

如何让程序在容器中运行起来：

把程序放在镜像中，通过镜像启动一个容器，容器启动时，会执行我们的程序

比如：

- 1、编写自己程序代码。
- 2、选择镜像，并下载到宿主机。

比如C代码可使用gcc g++ ubuntu centos 三种镜像，测试选择gcc

在192.168.1.29服务器上测试时，并没有安装docker，使用命令

apt-get install docker

apt-get install docker.io

安装成功

使用命令

docker search gcc

查看可选的gcc版本,一般用stars最多的那个

```
Processing triggers for ureadahead (0.100.0-16) ...
root@emic-Vostro-470 10:53:31:/home/zhangt/zt/https-github.com-miguelgrinberg-flasky/docker# docker search gcc
NAME                DESCRIPTION                STARS     OFFICIAL   AUTOMATED
gcc                  The GNU Compiler Collection is a compiling... 463       [OK]
rikorose/gcc-cmake  Build on top off the official gcc image in... 20
arm32v7/gcc          The GNU Compiler Collection is a compiling... 5
synapsedev/gcc-arm-none-eabi Docker build for gcc-arm-none-eabi usage i... 5
arm64v8/gcc          The GNU Compiler Collection is a compiling... 4
conanio/gcc8         1
ppc64le/gcc          The GNU Compiler Collection is a compiling... 1
reaverproject/gcc-boost Images with GCC and Boost built from sourc... 1
conanio/gcc6         1
teeks99/gcc-ubuntu   Versions of gcc running on ubuntu The goa... 0
celiangarcia/gcc7-cmake Omake built on top of official gcc 7 image. 0
amd64/gcc            The GNU Compiler Collection is a compiling... 0
rushmash/gcc-arm-embedded-docker gcc-arm-none-eabi toolchain 0
arm32v5/gcc          The GNU Compiler Collection is a compiling... 0
cyberdojofoundation/gcc_assert 0
celiangarcia/gcc8-cmake Omake built on top of official gcc 8 image. 0
conanio/gcc5         0
conanio/gcc7         0
coderunner/gcc       0
celiangarcia/gcc6-cmake Omake built on top of official gcc 6 image. 0
mattgodbolt/gcc-builder Builds GCCs for Compiler Explorer https://... 0
wearlifetrading/gcc5 Build for gcc5 development machine 0
yolo/gcc_source_resource abs 0
s390x/gcc            The GNU Compiler Collection is a compiling... 0
trollin/gcc         0
```

使用命令

docker pull gcc

下载镜像

注：第一次执行时，出错，报错FATA[0102] Could not reach any registry endpoint

错误原因是docker版本太低了

执行apt-get upgrade docker 进行了版本更新，提升了docker版本，后再次拉去成功

```
ot@emic-Vostro-470 14:25:29:/home/zhangt/zt/https-github.com-miguelgrinberg-flasky/docker# docker pull gcc
test: Pulling from gcc
a8ba3f75c8: Pull complete
3de42f0576: Pull complete
2702018869: Pull complete
45730237e4: Pull complete
947514bd4e: Pull complete
d4973a435e: Pull complete
cdc6662e3f: Pull complete
d5dfa2e00e: Pull complete
d5dfa2e00e: Pulling fs layer
bb0ba3f4f0: Already exists
fc7770b8d6: Already exists
4c23e109e1: Already exists
c78ccf0a09: Already exists
3265044fa2: Already exists
Digest: sha256:c566d22feb909de15d776b08085315198cc83db54d7b5e00fa20a0bd5d298a63
Status: Downloaded newer image for gcc:latest
ot@emic-Vostro-470 14:37:28:/home/zhangt/zt/https-github.com-miguelgrinberg-flasky/docker#
```

使用命令

docker images

查看已下载的镜像

3、制作自己的镜像

方式1：(直接运行镜像文件，生成容器，容器中修改内容，并保存镜像)

使用命令

docker run -ti gcc:latest /bin/bash

其中 gcc:latest 前面是仓库，后面是tag，最后的 /bin/bash是执行的脚本，也可以是sh,最好使用bash，bash可以直接看到id

```
#
# exit
root@emic-Vostro-470 15:39:32:/home/zhangt/zt/https-github.com-miguelgrinberg-flasky/docker# docker run -ti gcc:test /bin/bash
root@468919df47a6:/#
```

进入到docker容器中后，命令行中会显示容器id,ac1134211d6b, 可以执行 创建文件。比如创建一个test.txt文件，并保存。

然后

exit

退出

使用命令

docker commit -a="zhangt" -m="add a text.txt file for test" ac1134211d6b

gcc:test

其中 -a后是镜像的生成者， -m用于本次镜像提交的备注信息， ac1134211d6b 是进入到容器后的id, gcc是镜像仓库， test是本次的标注

备注：这里的commit和git相似，是推送到本次仓库中。push时可以选择自己的远程镜像仓库。默认的gcc肯定是不允许push的。

最后，使用

docker images

可以查看到新生成的docker 镜像

```
exit
root@emic-Vostro-470 15:44:01:/home/zhangt/zt/https-github.com-miguelgrinberg-flasky/docker# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
gcc                  test               ba645dab098a       11 minutes ago     1.136 GB
gcc                  latest            acd5dfa2e00e       4 weeks ago        1.136 GB
root@emic-Vostro-470 15:44:04:/home/zhangt/zt/https-github.com-miguelgrinberg-flasky/docker#
```

再次使用命令：

docker run -ti gcc:test /bin/bash

进入容器后，可以查看到上次生成text.txt文件

```
exit
root@emic-Vostro-470 15:44:01:/home/zhangt/zt/https-github.com-miguelgrinberg-flasky/docker# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
gcc                  test               ba645dab098a       11 minutes ago     1.136 GB
gcc                  latest            acd5dfa2e00e       4 weeks ago        1.136 GB
root@emic-Vostro-470 15:44:04:/home/zhangt/zt/https-github.com-miguelgrinberg-flasky/docker# docker run -ti gcc:test /bin/bash
root@50f053087741:/# ls
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys text.txt tmp usr var
root@50f053087741:/#
```

方法2：使用Dockerfile生成镜像

使用docker commit 命令的方式可以快速创建一个新镜像，但有些不便之处（啥不便之处呢？ ？ ？ ？ 说是不能共享开发过程）

首先创建一个Dockerfile，注意：文件名就是Dockerfile

内容如下：（具体Dockerfile的语法单独文档说明）

```
1 #this is comment
2 FROM gcc:latest
3 MAINTAINER zhangt <zhangt@emicnet.com>
4 RUN mkdir /usr/src/myapp/
5 COPY test.c /usr/src/myapp/
6 WORKDIR /usr/src/myapp
7 RUN gcc hello.c -o hello
8 CMD ["/hello"]
```

使用命令：

docker build -t zgcc:v1 .

构建一个镜像

上述命令中, gcc:v1 是镜像库和标记, 最后还有一个. 表示当前目录 (docker会在该目录下搜索Dockerfile, 另外, Dockerfile中会有拷贝文件等操作到容器中, 需要指明源地址) 编译通过后,

使用命令

docker images

查看镜像:

```
root@emic-Vostro-470 16:51:55:/home/zhangt/zt/https-github.com-miguelgrinberg-flasky/docker# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
zhangt/gcc           v1                 2fcb4611b08d       8 minutes ago      1.136 GB
gcc                  test               ba645dab098a       About an hour ago  1.136 GB
gcc                  latest             acd5dfa2e00e       4 weeks ago        1.136 GB
root@emic-Vostro-470 16:51:58:/home/zhangt/zt/https-github.com-miguelgrinberg-flasky/docker#
```

运行镜像:

docker run -d zhangt/gcc:v1

-d表示后台运行, 可以不加, 建议加上

运行后, 使用命令

docker ps

查看运行中的容器

```
root@emic-Vostro-470 17:08:02:/home/zhangt/zt/https-github.com-miguelgrinberg-flasky/docker# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
f0af65fa0d2e       zhangt/gcc:v1      "./hello"          3 minutes ago      Up 3 minutes                clever_fermat
root@emic-Vostro-470 17:09:30:/home/zhangt/zt/https-github.com-miguelgrinberg-flasky/docker#
```

可使用命令:

docker exec -it clever_fermat /bin/bash

进入到容器中, 可以在目录下看到t.txt

tail -f t.txt

可以看到程序在运行。

通过ps -eLf命令可以查看运行中的进程。可以看到./hello运行中

备注:

该方法中, 是将源代码拷贝到容器中, 在生成容器时编译, 容器运行时执行cmd指定的程序。

其实也可以在宿主机上直接编译好, 生成obj,直接拷贝目标文件到容器中, 直接运行。