docker入门教程

对于使用 systemd 的系统，请在 /etc/docker/daemon.json 中写入如下内容（如果文件不存在请新建该文件）

```
 "registry-mirrors":
"https://registry.docker-cn.com"
 ]
}
```

注意，一定要保证该文件符合 json 规范，否则 Docker 将不能启动。

之后重新启动服务。

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart docker
```

注意：如果您之前查看旧教程，修改了 docker.service 文件内容，请去掉您添加的内容（--registry-mirror=https://registry.docker-cn.com），这里不再赘述。

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.23.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker-compose --version
```

service配置

docker compose 配置文件

docker-compose.yml

```
version:  '3'
services:
  reverseproxy:
    build:
      dockerfile: Dockerfile
      context: .
    image:  nginx_1.14.2
    container_name: prod_reverse_proxy
    volumes:
      - /home/tpad/docker_samples/my_docker:/tmr
      - ./nginx/certs:/etc/nginx/certs
      - ./nginx/sites-available/:/etc/nginx/sites-
available
      - ./nginx/sites-enabled/:/etc/nginx/sites-enabled
      - ./nginx/private/:/etc/nginx/private
      - ./nginx/ssl:/etc/nginx/ssl
      - ./logrotate/nginx:/etc/logrotate.d/nginx
    ports:
      - 1014:1014
      - 443:443
      - 1002:1002
      - 1003:1003
      - 9000:9000
    logging:
      driver: "json-file"
      options:
        max-size: "1k"
        max-file: "3"
    healthcheck:
```

```
      test: ["CMD", "curl", "-f", "http://localhost:
1014"]
      interval: 5s
      timeout: 3s
      retries: 3
```

docker运行操作命令

docker run -it --rm --entrypoint="/bin/sh" ucucs/alpine-nginx:1.16.0

docker run -it --rm --entrypoint sh ucucs/alpine-nginx:1.16.0

docker run --name nginxhere --rm -d -p 80:80 ucucs/alpine-nginx:1.16.0

docker exec nginxhere nginx -t

docker-compose常用命令

和docker命令一样，docker-compose命令也有很多选项。下面我们来详细探讨docker-compose的常用命令。

build

构建或重新构建服务。服务被构建后将会以project*service* 的形式标记，例如：*composetest*db 。

help

查看指定命令的帮助文档，该命令非常实用。docker-compose所有命令的帮助文档都可通过该命令查看。

docker-compose help COMMAND

示例：

docker-compose help build # 查看docker-compose build的帮助

kill

通过发送SIGKILL 信号停止指定服务的容器。示例：

docker-compose kill eureka

该命令也支持通过参数来指定发送的信号，例如：

docker-compose kill -s SIGINT

logs

查看服务的日志输出。

port

打印绑定的公共端口。示例：

docker-compose port eureka 8761

这样就可输出eureka服务8761端口所绑定的公共端口。

ps

列出所有容器。示例：

docker-compose ps

也可列出指定服务的容器，示例：

docker-compose ps eureka

pull

下载服务镜像。

rm

删除指定服务的容器。示例：

docker-compose rm eureka

run

在一个服务上执行一个命令。示例：

docker-compose run web bash

这样即可启动一个web服务，同时执行bash命令。

scale

设置指定服务运行容器的个数，以service=num的形式指定。示例：

docker-compose scale user=3 movie=3

start

启动指定服务已存在的容器。示例：

docker-compose start eureka

stop

停止已运行的容器。示例：

docker-compose stop eureka

停止后，可使用docker-compose start 再次启动这些容器。

up

构建、创建、重新创建、启动，连接服务的相关容器。所有连接的服务都会启动，除非它们已经运行。

docker-compose up 命令会聚合所有容器的输出，当命令退出时，所有容器都会停止。

使用docker-compose up -d 可在后台启动并运行所有容器。

docker-compose执行命令特别慢的问题

yum install rng-tools

systemctl enable rngd

systemctl start rngd

发现异常请求有多次wait4、restart*syscall　resumed*等结果，观察到开头*open*了*/dev/random*，怀疑与此有关。

进行/dev/random测试发现果然有概率出现慢请求

主要原因是生成随机数的时候卡住了,导致tomcat启动不了。

是否有足够的熵来用于产生随机数，可以通过如下命令来查看

```
cat /proc/sys/kernel/random/entropy_avail
```

docker常用命令

docker命令详解

FROM

FROM指定一个基础镜像， 一般情况下一个可用的 Dockerfile一定是 FROM 为第一个指令。至于image则可以是任何合理存在的image镜像。

FROM 一定是首个非注释指令 Dockerfile.

FROM 可以在一个 Dockerfile 中出现多次，以便于创建混合的 images。

如果没有指定 tag ， latest 将会被指定为要使用的基础镜像版本。

MAINTAINER

这里是用于指定镜像制作者的信息

RUN

RUN命令将在当前image中执行任意合法命令并提交执行结果。命令执行提交后，就会自动执行Dockerfile中的下一个指令。

层级 RUN 指令和生成提交是符合Docker核心理念的做法。它允许像版本控制那样，在任意一个点，对image 镜像进行定制化构建。

RUN 指令缓存不会在下个命令执行时自动失效。比如 RUN apt-get dist-upgrade -y 的缓存就可能被用于下一个指令. --no-cache 标志可以被用于强制取消缓存使用。

ENV

ENV指令可以用于为docker容器设置环境变量

ENV设置的环境变量，可以使用 docker inspect命令来查看。同时还可以使用docker run --env <key>=<value>来修改环境变量。

USER

USER 用来切换运行属主身份的。Docker 默认是使用 root，但若不需要，建议切换使用者身分，毕竟 root 权限太大了，使用上有安全的风险。

WORKDIR

WORKDIR 用来切换工作目录的。Docker 默认的工作目录是/，只有 RUN 能执行 cd 命令切换目录，而且还只作用在当下下的 RUN，也就是说每一个 RUN 都是独立进行的。如果想让其他指令在指定的目录下执行，就得靠 WORKDIR。WORKDIR 动作的目录改变是持久的，不用每个指令前都使用一次 WORKDIR。

COPY

COPY 将文件从路径 <src> 复制添加到容器内部路径 <dest>。

<src> 必须是想对于源文件夹的一个文件或目录，也可以是一个远程的url，<dest> 是目标容器中的绝对路径。

所有的新文件和文件夹都会创建UID 和 GID 。事实上如果 <src> 是一个远程文件URL，那么目标文件的权限将会是600。

ADD

ADD 将文件从路径 <src> 复制添加到容器内部路径 <dest>。

<src> 必须是想对于源文件夹的一个文件或目录，也可以是一个远程的url。<dest> 是目标容器中的绝对路径。

所有的新文件和文件夹都会创建UID 和 GID。事实上如果 <src> 是一个远程文件URL，那么目标文件的权限将会是600。

VOLUME

创建一个可以从本地主机或其他容器挂载的挂载点，一般用来存放数据库和需要保持的数据等。

EXPOSE

EXPOSE 指令指定在docker允许时指定的端口进行转发。

CMD

Dockerfile.中只能有一个CMD指令。 如果你指定了多个，那么最后个CMD指令是生效的。

CMD指令的主要作用是提供默认的执行容器。这些默认值可以包括可执行文件，也可以省略可执行文件。

当你使用shell或exec格式时， CMD 会自动执行这个命令。

ONBUILD

ONBUILD 的作用就是让指令延迟執行，延迟到下一个使用 FROM 的 Dockerfile 在建立 image 时执行，只限延迟一次。

ONBUILD 的使用情景是在建立镜像时取得最新的源码（搭配 RUN）与限定系统框架。

ARG

ARG是Docker1.9 版本才新加入的指令。

ARG 定义的变量只在建立 image 时有效，建立完成后变量就失效消失

LABEL

定义一个 image 标签 Owner，并赋值，其值为变量 Name 的值。(LABEL Owner=$Name)

ENTRYPOINT

是指定 Docker image 运行成 instance（也就是 Docker container）时，要执行的命令或者文件。