2.Docker笔记 - 进阶

保存和共享镜像

docker commit 保存镜像 -m <提交内容>
docker tag 设置别名
也可以保存镜像时 设置别名 docker commit -m <提交内容> <镜像> <别名:版本>
docker save <别名:版本> > <文件名.tar> 迁移镜像
docker save -o <文件名.tar> <别名:版本> <...> 这样也能迁移 能同时导出多个
docker load < <文件名.tar> 导入镜像
docker load -i <文件名.tar> 这样也能导入
docker export -o <文件名.tar> <别名:版本> 导出容器 保存运行中的镜像再导出 commit save 的合体
Docker import <文件名.tar> <别名:版本> 导入容器 实际是导入镜像

DockerFile

来源

直接创建一个文件名为Dockerfile的文件
FROM <image> [AS <name>]
FROM <image>[:<tag>] [AS <name>]
FROM <image>[@<digest>] [AS <name>] 指定基础镜像

执行命令

构建时执行 RUN

RUN <command>
RUN ["executable", "param1", "param2"] 执行命令并创建新的镜像层
支持 \ 换行 所以不需要创建多个镜像层时 最好一行写完run run是在构建镜像时跑完

启动运行时的ENTRYPOINT和CMD

ENTRYPOINT	CMD	实际执行
ENTRYPOINT ["/bin/ep", "arge"]		/bin/ep arge
ENTRYPOINT /bin/ep arge		/bin/sh –c /bin/ep arge
	CMD ["/bin/exec", "args"]	/bin/exec args
	CMD /bin/exec args	/bin/sh –c /bin/exec args
ENTRYPOINT ["/bin/ep", "arge"]	CMD ["/bin/exec", "argc"]	/bin/ep arge /bin/exec argc
ENTRYPOINT ["/bin/ep", "arge"]	CMD /bin/exec args	/bin/ep arge /bin/sh –c /bin/exec args
ENTRYPOINT /bin/ep arge	CMD ["/bin/exec", "argc"]	/bin/sh –c /bin/ep arge /bin/exec argc
ENTRYPOINT /bin/ep arge	CMD /bin/exec args	/bin/sh –c /bin/ep arge /bin/sh –c /bin/exec args

CMD ["executable","param1","param2"] 不会自带sh
CMD command param1 param2
当run启动没有写命令 DockerFile又没有ENTRYPOINT时 CMD是默认当命令行运行 CMD只有一个才有效 当有多个时 最后一个生效

ENTRYPOINT ["executable", "param1", "param2"] 用中括号会忽略全部cmd的参数 并且不会自带sh
ENTRYPOINT command param1 param2

ENTRYPOINT目的是容器启动时的初始化
CMD目的是启动命令
ENTRYPOINT比CMD优先级高 它们两则只能存在一个 如果都存在 一定是执行最后一行ENTRYPOINT
如果两者都存在 在ENTRYPOINT非中括号情况下 CMD会成为ENTRYPOINT的属性

暴露端口

EXPOSE <port> [<port>/<protocol>...] 端口暴露 其他容器通过 --link 选项连接

数据卷

VOLUME ["/data"] 自动建立或使用数据卷

复制拷贝
COPY [--chown=<user>:<group>] <src>... <dest>
ADD [--chown=<user>:<group>] <src>... <dest>

有空格时 使用以下方法
COPY [--chown=<user>:<group>] ["<src>",... "<dest>"]
ADD [--chown=<user>:<group>] ["<src>",... "<dest>"]
ADD可以用远程url 如果是压缩包 还会自动解压

构造镜像
docker build <保存路径> 构建镜像 -t <别名:版本> -f <指定Dockerfile目录> --no-cache构建时 禁用镜像缓存

变量
ARG **变量 占位符** docker build构建时候 可以通过--build-arg xx=xx传参 引用时候 前面加$ 构建时可用
ENV 环境变量 docker run启动运行时可以通过-e --env xx=xx覆盖环境变量 **引用一样前面加$ 构建和创建时 都可用**
ENV比ARG优先级高 同名情况下 无论循序如何 ENV都会覆盖ARG