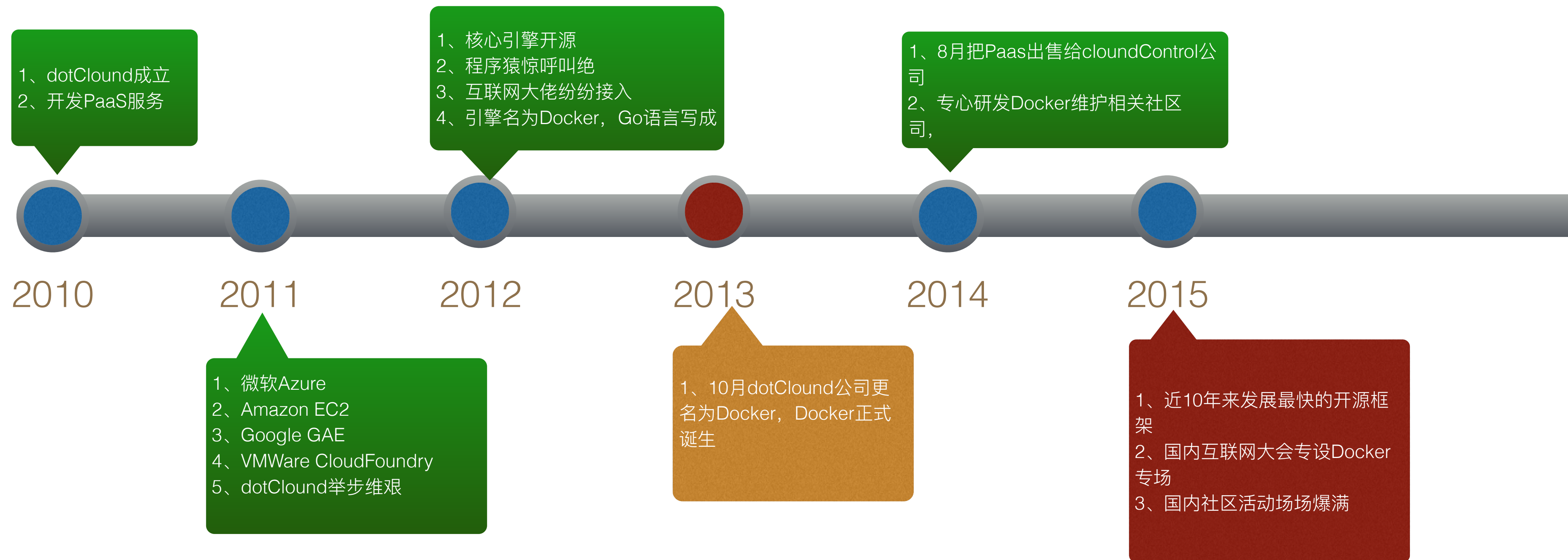




# Docker入门与实践

## 第一章：基础篇



# 我这里好好的，为什么到你那里不行了？

Docker解决了这个问题

程序跑起来的几个要素：

代码   依赖服务   运行环境   配置

如果代码和依赖服务相同，那么程序跑不起来的原因是：

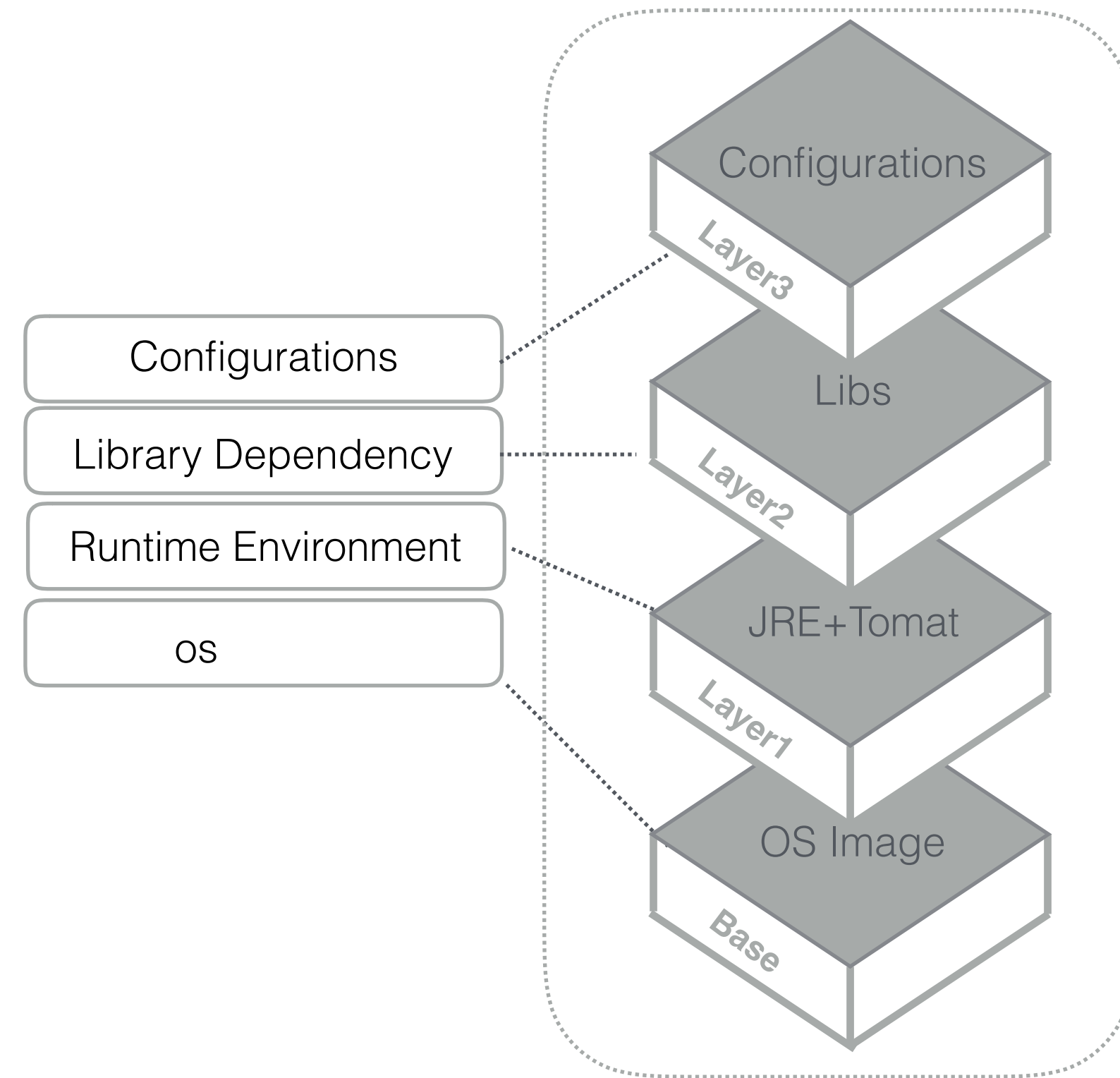
运行环境   配置

这二货是元凶

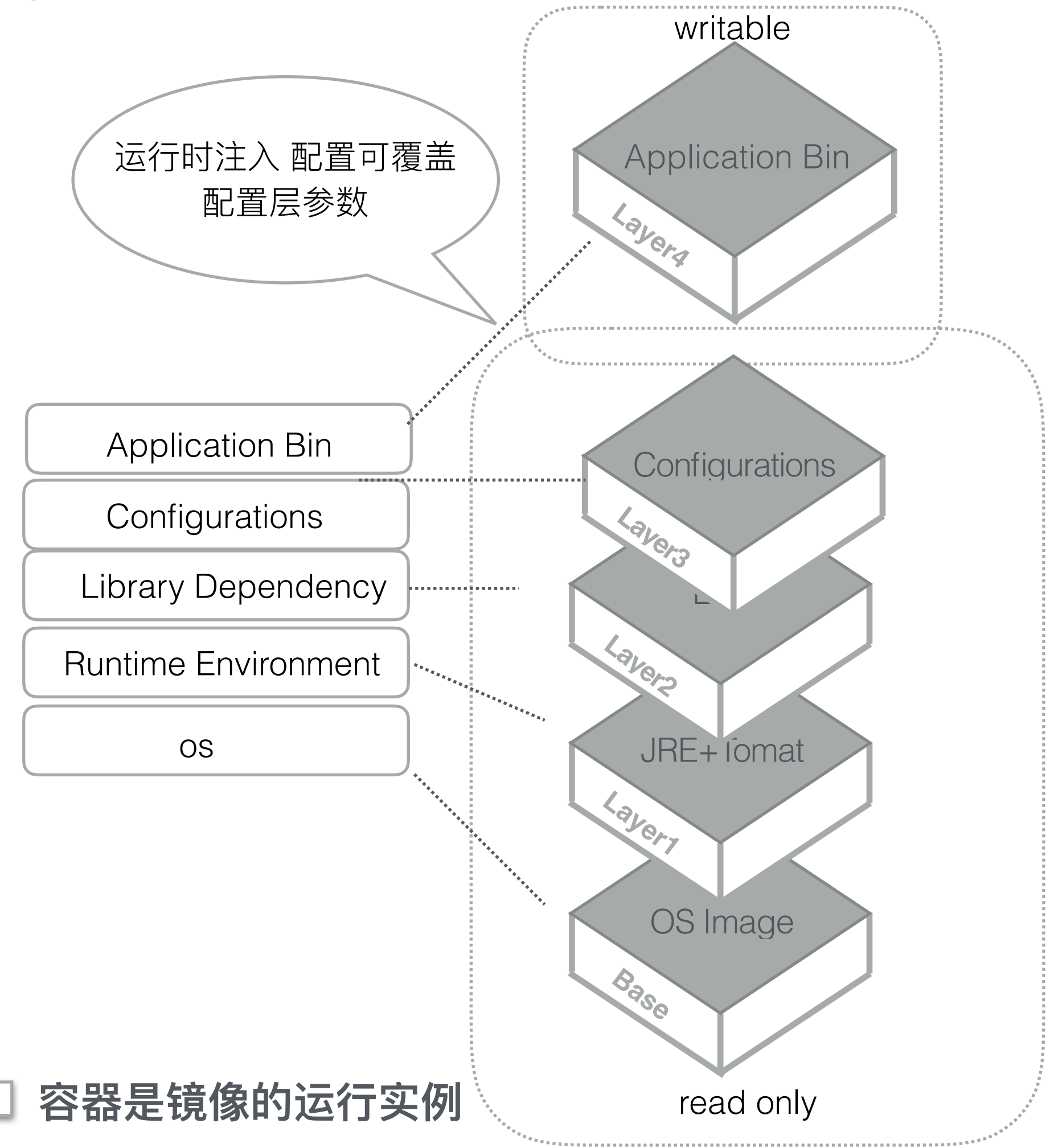
# Docker镜像



# Docker容器



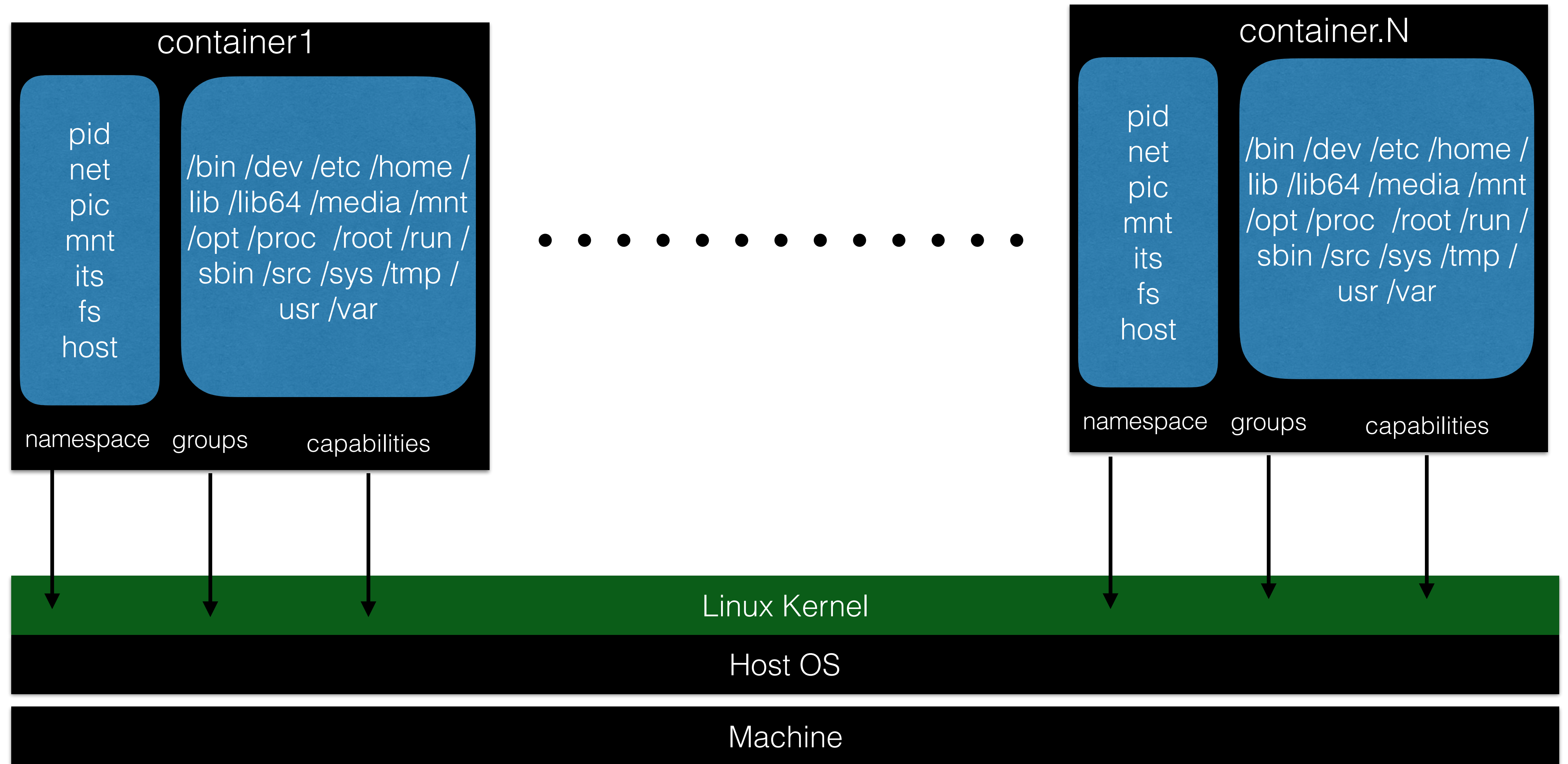
- ☐ 镜像是个特殊的模版目录
  - ☒ 可传递
  - ☒ 可版本管理
  - ☒ 分层的
  - ☒ 每层内容可缓存
  - ☒ 每层内容可索引
- ☐ 镜像用来创建容器



- ☐ 容器是镜像的运行实例
- ☐ 应用通过容器运行
  - ☒ 启动时,在所有镜像层之上创建一层可写层
  - ☒ 镜像只是读的
- ☐ 容器可转为镜像



# 什么是容器



共享Linux内核与系统中其它进程资源隔离的类操作系统的轻量级进程

# 镜像的构建

## ☐ 两种构建方式

### — 通过容器构建镜像

构建命令：`docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]`

### — 通过Dockerfile文件构建镜像

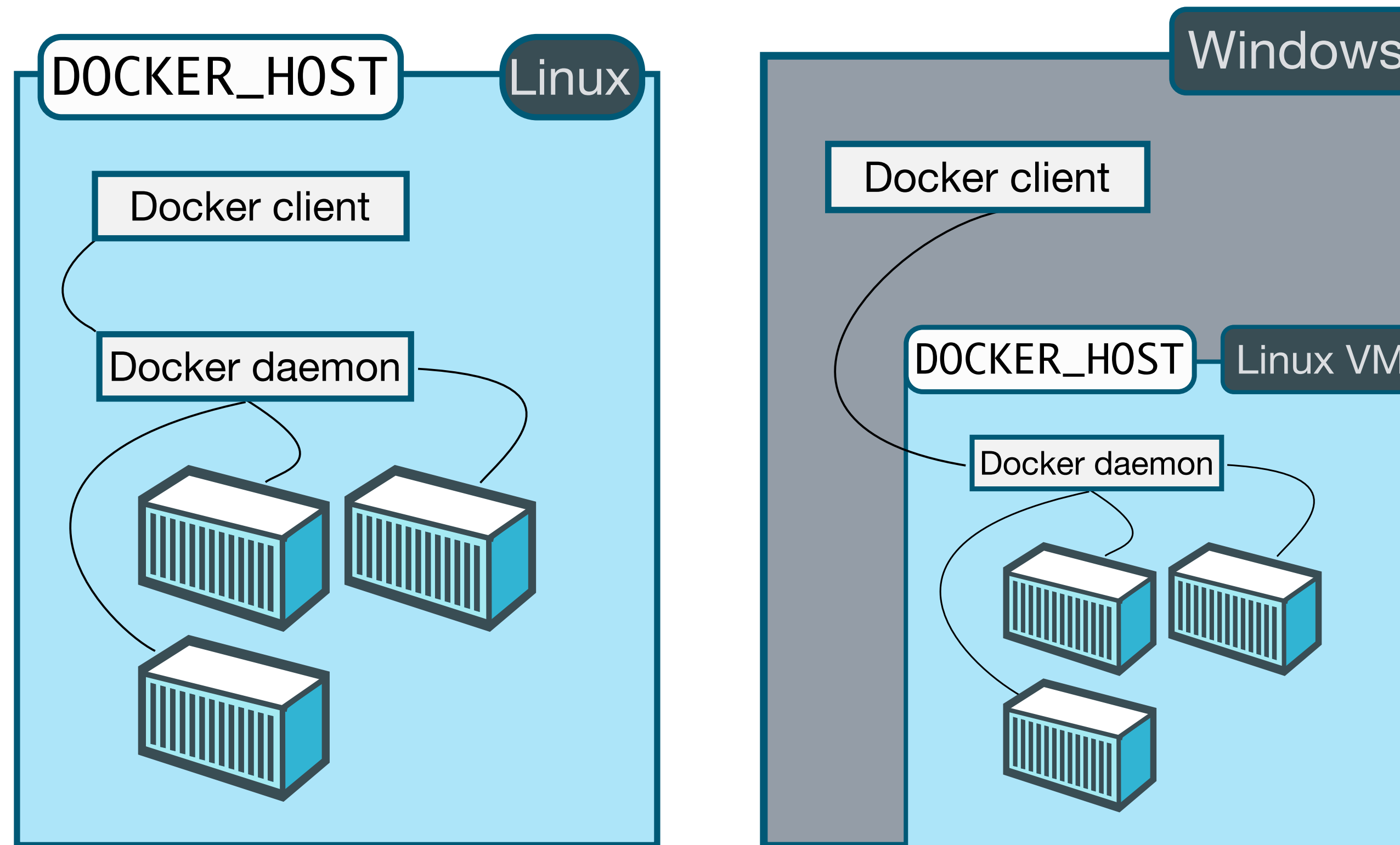
构建命令：`docker build -t 'image tag name' .`

## ☐ 作用

### — 自定义镜像的能力

### — 以软件的形式打包并分发服务及其运行环境

# Docker Client & Docker Daemon



## ☐ Docker Client

### ☐ 作用

向Docker daemon发起请求，执行相应的管理操作

### ☐ 存在

可以是命令行工具

可以是任何遵循Docker API的客户端

## ☐ Docker Daemon

### ☐ 作用

负责响应来自Docker Client的请求，  
将请求翻译成系统调用完成容器管理操作

### ☐ 存在

Docker最核心的后台进程

# Docker仓库

## □ 集中存放镜像文件的场所

### — 仓库

- 公开仓库: **Docker Hub**
- 私有仓库: **Private Docker Registry**

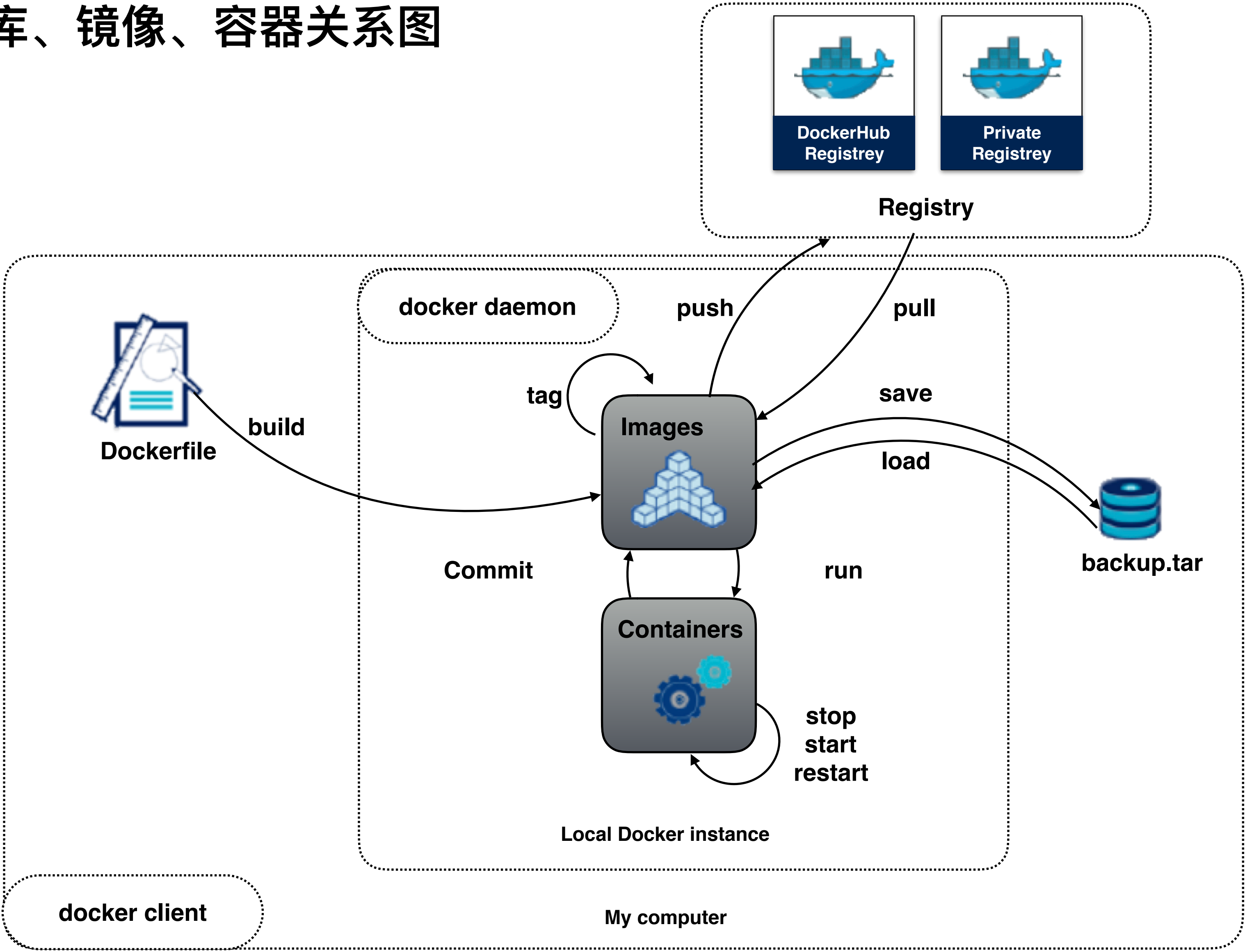
### — 仓库操作

- **Docker 中心创建帐号**
- **登陆仓库: docker login**    注: 你的身份验证凭证将被存储在你本地目录的.dockercfg文件中。
- **搜索镜像: docker search**    注: 可对镜像名称、用户名或者描述等进行搜索;

\*注: Docker 仓库的概念跟 Git 类似, 注册服务器可以理解为 GitHub 这样的托管服务。



# 仓库、镜像、容器关系图



VM

VS

Docker

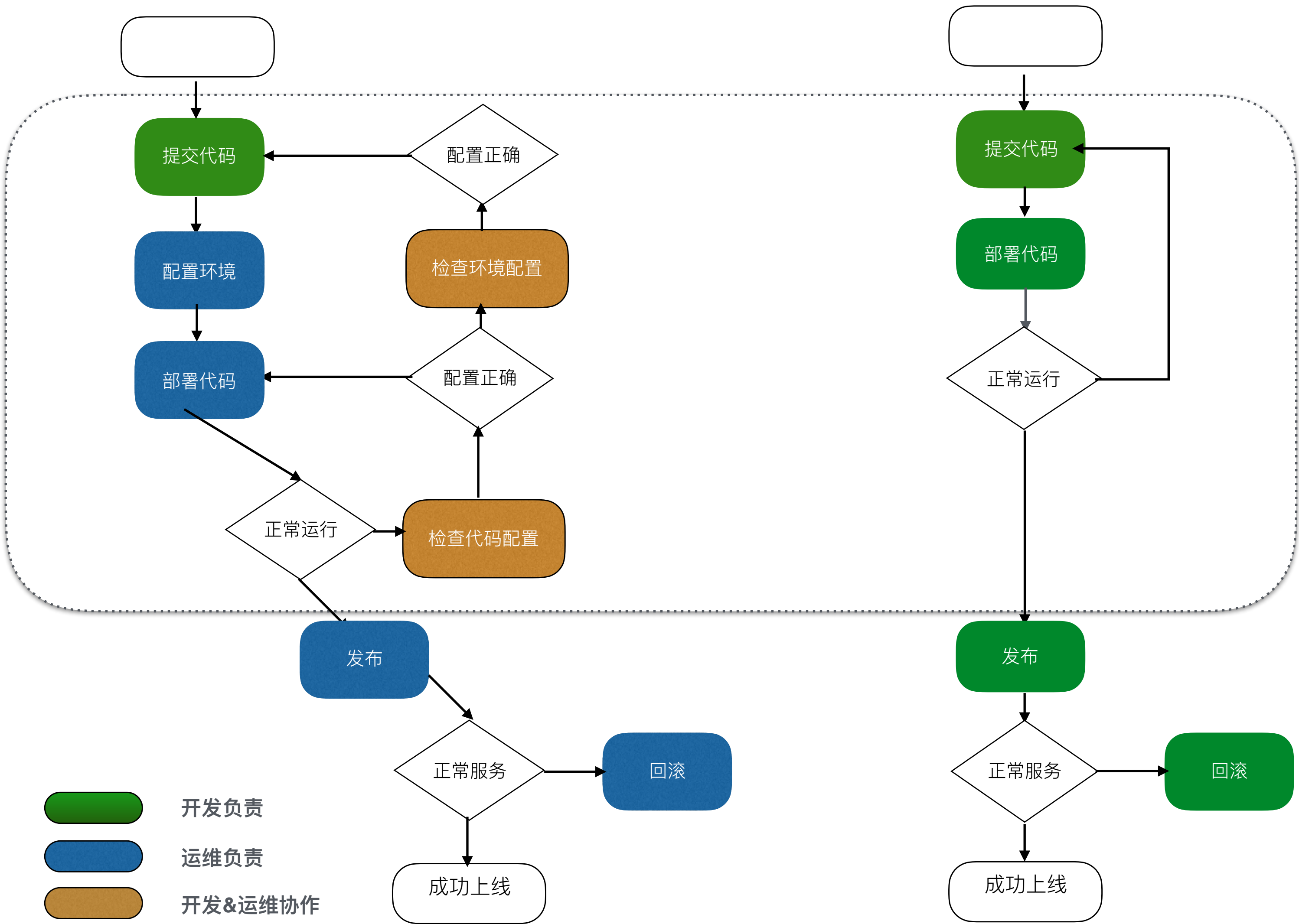
对比项	容器技术	虚拟机技术
占用磁 盘空间	小，甚至几十KB(镜像层情况)	非常大，上GB
启动速度	快，几秒种	慢，几分钟
运行状态	运行于宿主机的内核， 容器共享同一个Linux内核	运行于Hypervisor上
并发性	一台宿主机可以启动成千上百个容器	最多几十个虚拟机
性能	接近宿主机本地进程	逊于宿主机
IO/CPU/MEM资源利用率	高	低

天下武功，唯快不破 => Docker完胜

运维&开发驱动

VS

开发驱动



开发驱动 => 简化流程降低协作成本

# Docker安装

## ☐ Docker安装主要参考官网

### — Mac OS X下安装

- Docker Toolbox
- <https://docs.docker.com/engine/installation/mac>

### — Windwos下安装

- Docker Toolbox
- <https://docs.docker.com/engine/installation/windows/>

### — Debian下安装

- Docker Toolbox
- <https://docs.docker.com/engine/installation/debian/>

# 镜像常用操作命令

## ☐ 拉取镜像

— Usage: `docker pull [OPTIONS] NAME[:TAG|@DIGEST]`

```
hujiabaos-MBP:~ hujiabao$ docker pull mysql:5.7.10
5.7.10: Pulling from library/mysql
523ef1d23f22: Pulling fs layer
140f9bdfef97: Pulling fs layer
2df110ab8d10: Pulling fs layer
8ce18aa992e7: Pulling fs layer
19aa914a4bb3: Pulling fs layer
46a35d16da7c: Pulling fs layer
60ce6e65025d: Pulling fs layer
56080edc0174: Pulling fs layer
e7be56140f8b: Pulling fs layer
3666f062674e: Pulling fs layer
3c3357b44a48: Pulling fs layer
bd3d56d4cb20: Pulling fs layer
01e70da302a5: Pulling fs layer
d7ef45ff532e: Pulling fs layer
8eb845d8d7f6: Pulling fs layer
d39c3fa09ced: Pulling fs layer
```



# 镜像常用操作命令

☐ 列出本地镜像

```
— docker images [OPTIONS] [REPOSITORY[:TAG]]
```

```
hujiabaos-MacBook-Pro:docker-training hujiabao$ docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
aaa                      latest             b0e41ed4cce4        About an hour ago   652.7 MB
<none>                   <none>             3afe50041b65        About an hour ago   652.7 MB
mesql                    5.7.10             d9f509654942        7 hours ago         360.3 MB
node                     latest             3a18b51160d3        13 days ago         643.1 MB
mysql                    5.7.10             d39c3fa09ced        2 weeks ago         360.3 MB
easynode                 0.0.3              e52f92564758        2 weeks ago         698.4 MB
registry.hz.netease.com/easynode 0.0.3              e52f92564758        2 weeks ago         698.4 MB
```

# 镜像常用操作命令

## □ 运行镜像

— Usage: `docker run [OPTIONS] IMAGE [COMMAND] [ARG...]`

```
hujiabaos-MBP:~ hujiabao$ docker run --name mysql3 -e MYSQL_ROOT_PASSWORD='aa' mysql
Initializing database
2016-01-27T16:19:04.723677Z 0 [Warning] InnoDB: New log files created, LSN=45790
2016-01-27T16:19:04.761579Z 0 [Warning] InnoDB: Creating foreign key constraint system tables.
2016-01-27T16:19:04.826476Z 0 [Warning] No existing UUID has been found, so we assume that this is the first time that this server has been
2016-01-27T16:19:04.828323Z 0 [Warning] Gtid table is not ready to be used. Table 'mysql.gtid_executed' cannot be opened.
2016-01-27T16:19:04.829073Z 1 [Warning] root@localhost is created with an empty password ! Please consider switching off the --initialize-
2016-01-27T16:19:05.278181Z 1 [Warning] 'user' entry 'root@localhost' ignored in --skip-name-resolve mode.
2016-01-27T16:19:05.278741Z 1 [Warning] 'user' entry 'mysql.sys@localhost' ignored in --skip-name-resolve mode.
2016-01-27T16:19:05.279259Z 1 [Warning] 'db' entry 'sys mysql.sys@localhost' ignored in --skip-name-resolve mode.
2016-01-27T16:19:05.279535Z 1 [Warning] 'proxies_priv' entry '@ root@localhost' ignored in --skip-name-resolve mode.
2016-01-27T16:19:05.279932Z 1 [Warning] 'tables_priv' entry 'sys_config mysql.sys@localhost' ignored in --skip-name-resolve mode.
Database initialized
MySQL init process in progress...
2016-01-27T16:19:06.826662Z 0 [Note] mysqld (mysqld 5.7.10) starting as process 39 ...
2016-01-27T16:19:06.830857Z 0 [Note] InnoDB: PUNCH HOLE support available
2016-01-27T16:19:06.831025Z 0 [Note] InnoDB: Mutexes and rw_locks use GCC atomic builtins
2016-01-27T16:19:06.831559Z 0 [Note] InnoDB: Uses event mutexes
2016-01-27T16:19:06.831958Z 0 [Note] InnoDB: GCC builtin __atomic_thread_fence() is used for memory barrier
2016-01-27T16:19:06.832256Z 0 [Note] InnoDB: Compressed tables use zlib 1.2.8
2016-01-27T16:19:06.832590Z 0 [Note] InnoDB: Using Linux native AIO
```



# 镜像常用操作命令

## ☐ 删除镜像

— `docker rmi [OPTIONS] IMAGE [IMAGE...]`

```
hujiabaos-MacBook-Pro:~ hujiabao$ docker rmi -f 902
Deleted: 902740558c7ffa198bca4aa0de8b18410ef452a40bb0bb2f0fdad2854a440958
Deleted: fc9042b635f6036b30ae36db1a24685563d7ee4281f4f328bcea8f9828fabe8f
```

# 容器常用操作命令

## □ 启动容器

— Usage: `docker run [OPTIONS] IMAGE [COMMAND] [ARG...]`

- t 选项让Docker分配一个伪终端（pseudo-tty）并绑定到容器的标准输入上，
- i 则让容器的标准输入保持打开

```
hujiabaos-MacBook-Pro:~ hujiabao$ docker run -t -i mysql /bin/bash
root@1486897f0658:/#
```

# 容器常用操作命令

## ☐ 进入容器

— Usage: docker exec -it [containerId|containerName] /bin/bash

```
Error response from daemon: No such id: db95  
root@zhapn1:/home/hjb/scripts# docker exec -it db95 /bin/bash  
root@db95bc8e5b3a:/#
```



# 容器常用操作命令

查看容器

Usage: docker ps -a

```
hujiabaos-MacBook-Pro:docker-training hujiabao$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
7df154e0c092   mesql:5.7.10   "/entrypoint.sh mysql"  19 minutes ago Up 3 seconds  3306/tcp     mysql3
```

# 容器常用操作命令

## ☐ 删除容器

— `docker rm [OPTIONS] CONTAINER [CONTAINER...]`

```
hujiabaos-MacBook-Pro:docker-training hujiabao$ docker rm -f c7670
c7670
```

# 容器常用操作命令

## ☐ 容器转为镜像

— Usage: `docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]`

```
hujiabaos-MacBook-Pro:books hujiabao$ docker commit -m "hello, mysql" -a "hujiabao" 6cb mesql:5.7.10
d9f5096549424db6b5fa876f3b4dc34d1438e7430e801d22c742ac225be74dc2
```



# Dockerfile文件

## ☐ 作用

- 构建镜像
- 共享、快速分发构建环境
- 跨平台构建环境

## ☐ 文件格式

- 文本文件，包含自定义的指令和格式
- 所有指令
  - FROM、MAINTAINER、RUN、COPY、ADD、ENV、ENTRYPOINT、CMD、USER、EXPOSE、WORKDIR、ONBUILD
- 除FROM，每一指令生成一镜像层，一层一层叠加
- 镜像层是有缓存的，利用这一特性可以加快构建速度

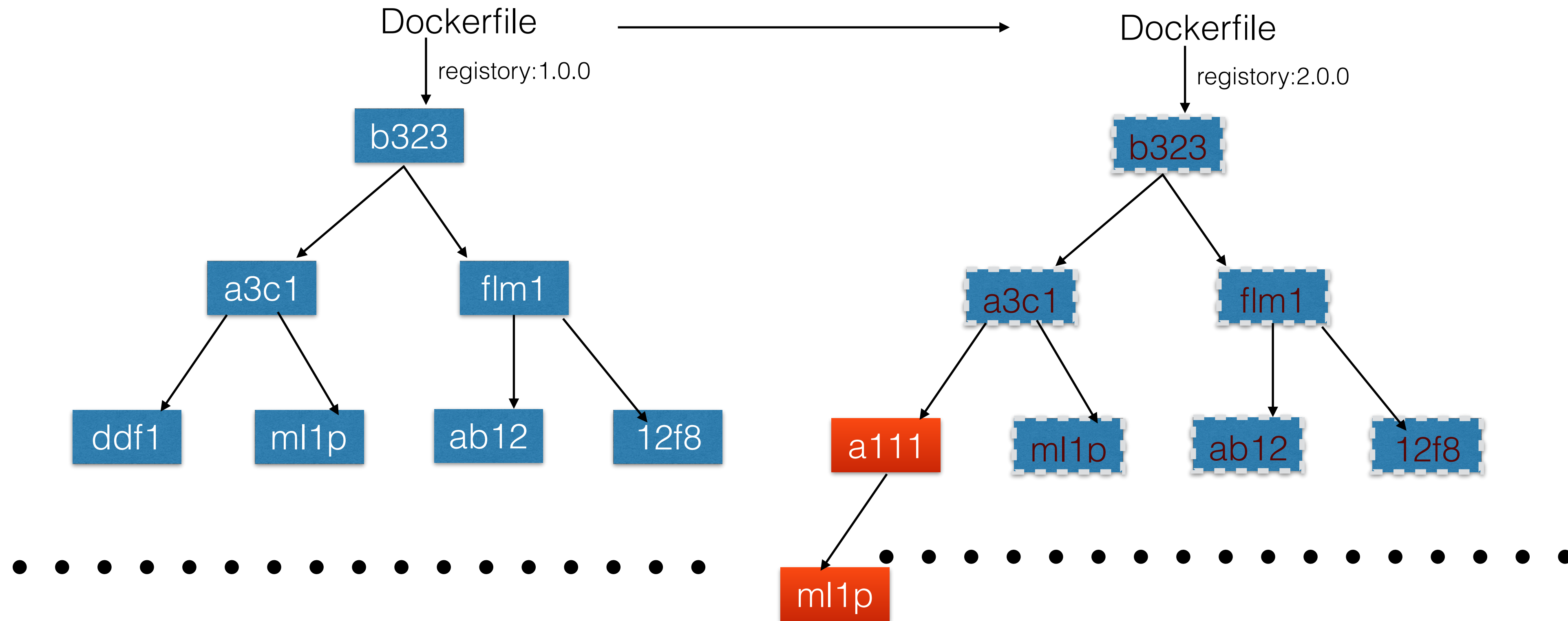
```
Sending build context to Docker daemon 9.605 MB
Step 1 : FROM node
---> 3a18b51160d3
Step 2 : MAINTAINER hujiabao
---> Using cache
---> 76c63902e479
Step 3 : RUN mkdir -p /usr/src/app
---> Using cache
---> 537024959bf3
Step 4 : COPY . /usr/src/app
---> ca0d56b49800
Removing intermediate container 49dd30eb0868
Step 5 : ENV VERSION 1.0.0
---> Running in 23adda5d2c02
---> 987d3d3e5625
Removing intermediate container 23adda5d2c02
Step 6 : RUN npm install
---> Running in b43cb498411b
npm info ok
---> b375c5001dd1
```

.....



# Dockerfile原理

- 像代码版本管理一样，管理镜像版本





# Dockerfile关键指令释义

## ❑ FROM

**语法：** FROM <image> [:<tag>]

**解释：** 设置要制作的镜像基于哪个基础镜像，FROM指令必须是整个Dockerfile的第一个指令，如果指定的镜像在本地不存在默认会自动从Docker Hub上下载

## ❑ MAINTAINER

**语法：** MAINTAINER <name>

**解释：** MAINTAINER 指令允许你给将要制作的镜像设置作者信息，一般作为第二条指令

## ❑ RUN

**语法：** RUN <command> or RUN ["executable", "param1", "param2"]

**解释：** RUN指令会在一个新的容器中执行任何命令

## ❑ CMD

**语法：** CMD ["execute" "param1","param2"] or CMD <command> [param1|param2] or CMD [param1|param2]

**解释：** CMD指令中指定的命令会在镜像运行时执行，只允许最后一条CMD指令有效,如果有ENTRYPOINT指令，则为其提供参数

# Dockerfile关键指令释义

## ☐ ADD

语法： ADD <src> <dst>

解释： 用于从指定路径拷贝一个文件或目录到容器的指定路径中，src可是文件或目录或URL

## ☐ COPY

语法： COPY <src> <dst>

解释： 用法与ADD相同，但不支持URL

## ☐ ENTRYPOINT

语法： ENTRYPOINT ["exe","para1","para2"] or ENTRYPOINT command para1 para2

解释： 指定镜像运行时运行的命令，最后一条有效，参数可以被docker run参数覆盖

## ☐ VOLUME

语法： VOLUME ["path"]

解释： 设置一个挂载点

## ☐ ENV

语法： ENV <key> <vlaue>

解释： 设置环境变量

# Dockerfile关键指令释义

## ❑ USER

语法： USER [username|uid]

解释： 设置用户或UID来运行生成的镜像和执行RUN指令

## ❑ WORKDIR

语法： WORKDIR /path/to/workdir

解释： 设置当前工作目录，默认为/目录

## ❑ ONBUILD

语法： ONBUILD [INSTRUCTION]

解释： 设置一些触发的指令,用于在当该镜像被作为基础镜像来创建其他镜像时执行一些操作

详见参考：<https://hujb2000.gitbooks.io/docker-flow-evolution/content/cn/basis/dockerfiledetail.html>

# Dockerfile使用举例

FROM node@5.5.0-wheezy

# 基于哪个基础镜像构建

MAINTAINER hujiabao

# 设置作者信息

RUN mkdir -p /usr/src/app

# 运行命令建立目录

COPY . /usr/src/app

# 拷贝宿主机当前目录到容器/usr/src/app目录

ENV VERSION 1.0.0

# 设置环境亦是VERSION=1.0.0

RUN npm install

# 安装依赖包

WORKDIR /usr/src/app/netease/bin

# 设置当前工作目录

USER node

# 设置用户或uid来运行生成的镜像和执行RUN指令

EXPOSE 80

# 告诉Docker daemon这个容器在运行时会监听哪些端口

ONBUILD echo 'I lurking in the parent image'

# 设计父镜像时，在子镜像构建时被触发的父镜像中的指令

ENTRYPOINT ["/start.sh"]

# 容启动时执行的最后一条命令

# Dockerfile使用建议

反对	建议	理由
<code>FROM node</code> 或 <code>FROM node@latest</code>	<code>FROM node@5.5.0-wheezy</code>	Dockerfile基于版本管理的，必须有明确的tag
<code>FROM node@5.5.0-wheezy</code> <code>COPY . /usr/src</code> <code>MAINTAINER netease-hjb</code>	<code>FROM node@5.5.0-wheezy</code> <code>MAINTAINER netease-hjb</code> <code>COPY . /usr/src</code>	利用一指令生成一镜像层的原理，不变的部分放在前面，
<code>ENV mysqlconfig 'http://***.json'</code>	<code>docker run --env mysqlconfig 'http://***.json'</code>	改变运行环境不用改变镜像
直接把日志写到容器里	<code>docker run -v /log:/log</code>	日志数据单独存放于数据容器或宿主机，不用存放于应用容器中
Dockerfile文件带ONBUILD指令 <code>docker build -t "hujb2000/node:v1.0.0" .</code>	Dockerfile文件带ONBUILD指令 <code>docker build -t "hujb2000/node:v1.0.0-ONBUILD" .</code>	带ONBUILD指令的必须给出提醒，以示后来者小心
<code>docker build -t "hujb2000/node" .</code>	<code>docker build -t "hujb2000/node:tag" .</code>	Dockerfile基于版本管理的，必须有明确的tag



谢谢

THANK YOU