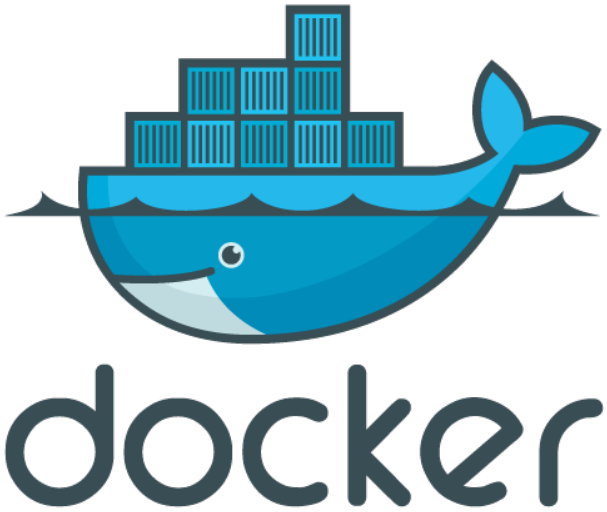


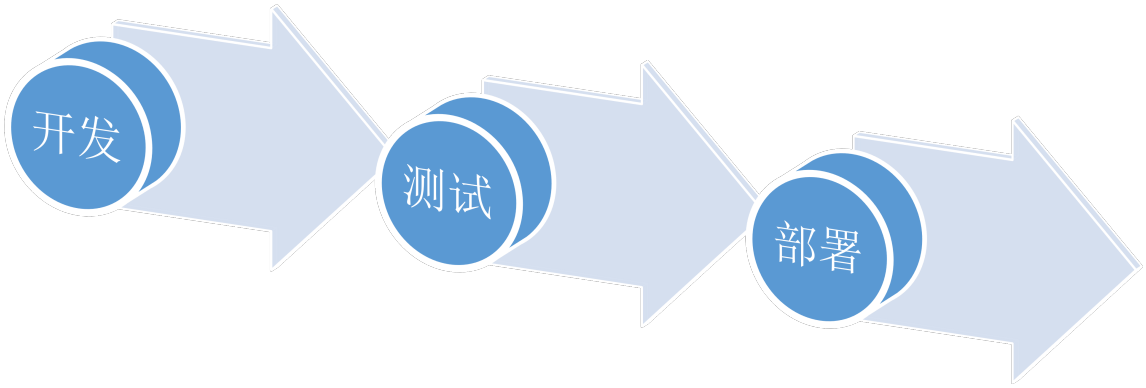
Introduction to Docker

杨润青 浙江大学

2017

是否曾经...

- 繁琐的开发环境搭建过程（换电脑 / 新入职）
- “代码在我机器能运行”
- 开发、测试、部署环境的不一致性



docker

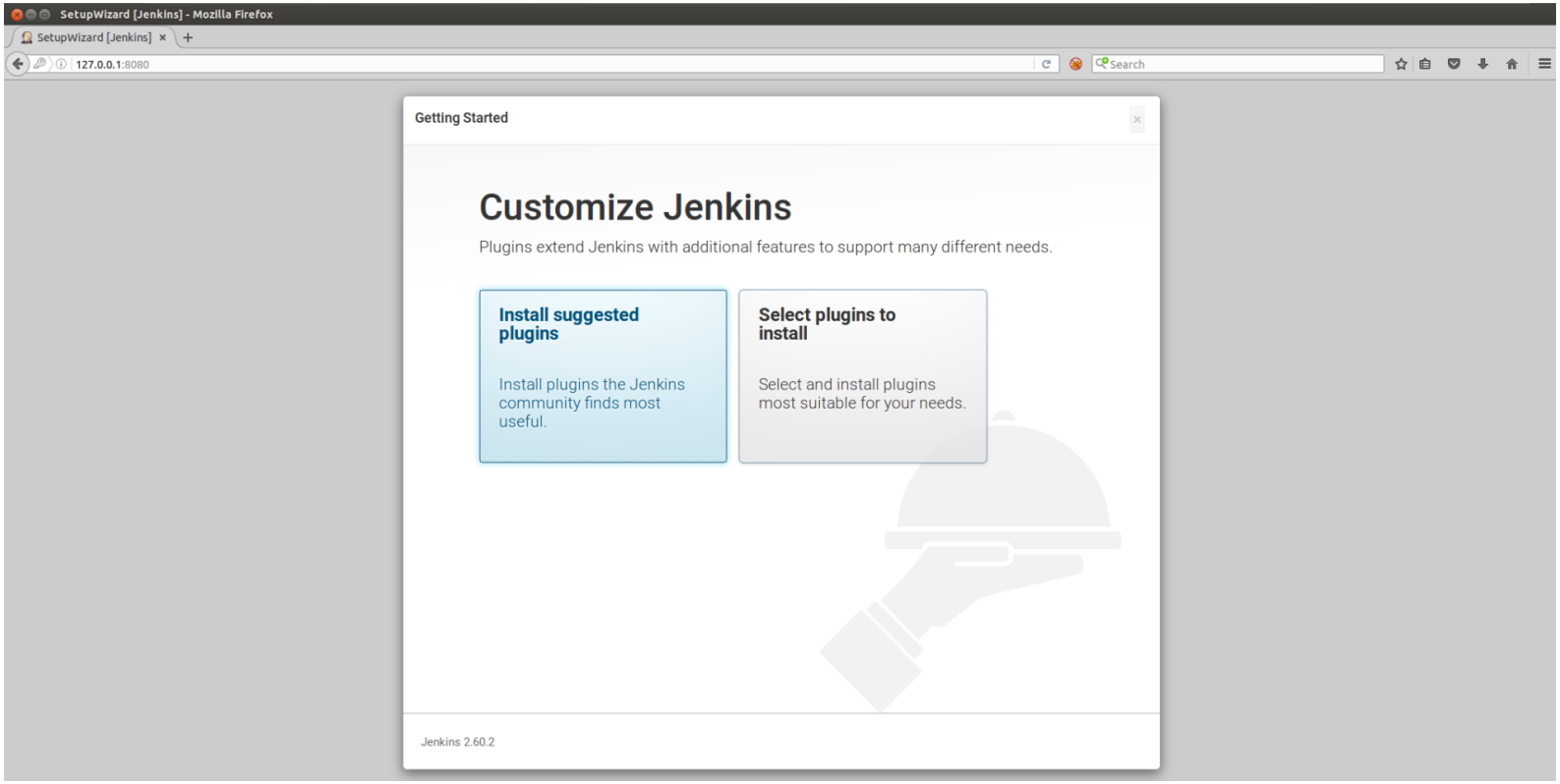
初识Docker

- 操作系统层面的虚拟化技术
- 对进程进行资源隔离 -> 容器！
  - > 独立的文件系统
  - > 独立的网络
  - > 独立的用户
  - > 。。。
- 将程序运行所需要的依赖和配置全部打包在了容器中！

使用Docker运行一个应用

- docker run –p 8080:8080 Jenkins

  不用任何安装/编译，直接可以运行



使用Docker运行一个应用

- docker run –ti –p 8080:8080 Jenkins /bin/bash

```
runqing@List-Cat:/$ ls -l
total 104
drwxr-xr-x   2 root root  4096 Oct 25  2016 bin
drwxr-xr-x   5 root root  4096 Feb 19  2017 boot
drwxr-xr-x  15 root root  4280 Sep  2 11:55 dev
drwxr-xr-x 183 root root 12288 Sep 12 13:27 etc
drwxr-xr-x  22 root root  4096 Sep 11 23:50 home
lrwxrwxrwx   1 root root    33 Oct 21  2015 initrd.img -> boo
lrwxrwxrwx   1 root root    33 Jul 10  2015 initrd.img.old ->
drwxr-xr-x  28 root root  4096 Feb 19  2017 lib
drwxr-xr-x   2 root root  4096 Nov  8  2016 lib32
drwxr-xr-x   2 root root  4096 Aug 24  2016 lib64
drwxr-xr-x   2 root root  4096 Aug 24  2016 libx32
drwx------   2 root root 16384 Jun 20  2015 lost+found
drwxr-xr-x   3 root root  4096 Apr 19 16:53 media
drwxr-xr-x   2 root root  4096 Apr 11  2014 mnt
drwxr-xr-x   4 root root  4096 Sep 14  2015 opt
dr-xr-xr-x 671 root root     0 Aug 20 11:02 proc
lrwxrwxrwx   1 root root    39 Jul  4 13:33 repo -> /home/tia
drwx------  15 root root  4096 Jul 22 22:54 root
drwxr-xr-x  38 root root  1280 Sep 14 20:01 run
drwxr-xr-x   2 root root 12288 Sep  8 11:37 sbin
drwxr-xr-x   3 root root  4096 Sep 15  2016 srv
dr-xr-xr-x  13 root root     0 Sep  8 10:45 sys
drwxr-xr-x   2 root root  4096 Jul  4 13:21 texbucket
drwxrwxrwt  22 root root  4096 Sep 14 20:03 tmp
drwxr-xr-x  13 root root  4096 Aug 24  2016 usr
drwxr-xr-x  14 root root  4096 Jul  4 12:54 var
lrwxrwxrwx   1 root root    30 Oct 21  2015 vmlinuz -> boot/v
lrwxrwxrwx   1 root root    30 Jul 10  2015 vmlinuz.old -> bo
```

主机

```
jenkins@288a0efc8cb5:/$ ls -l
total 48
lrwxrwxrwx   1 root root    7 Jul 23 00:00 bin -> usr/bin
drwxr-xr-x   2 root root 4096 Jul 13 13:04 boot
drwxr-xr-x   5 root root  360 Sep 14 12:03 dev
lrwxrwxrwx   1 root root   33 Jul 24 18:09 docker-java-home -> /usr/li
drwxr-xr-x  72 root root 4096 Sep 14 12:03 etc
drwxr-xr-x   2 root root 4096 Jul 13 13:04 home
lrwxrwxrwx   1 root root    7 Jul 23 00:00 lib -> usr/lib
lrwxrwxrwx   1 root root    9 Jul 23 00:00 lib32 -> usr/lib32
lrwxrwxrwx   1 root root    9 Jul 23 00:00 lib64 -> usr/lib64
lrwxrwxrwx   1 root root   10 Jul 23 00:00 libx32 -> usr/libx32
drwxr-xr-x   2 root root 4096 Jul 23 00:00 media
drwxr-xr-x   2 root root 4096 Jul 23 00:00 mnt
drwxr-xr-x   2 root root 4096 Jul 23 00:00 opt
dr-xr-xr-x 671 root root    0 Sep 14 12:03 proc
drwx------   2 root root 4096 Jul 23 00:00 root
drwxr-xr-x   4 root root 4096 Jul 23 00:00 run
lrwxrwxrwx   1 root root    8 Jul 23 00:00 sbin -> usr/sbin
drwxr-xr-x   2 root root 4096 Jul 23 00:00 srv
dr-xr-xr-x  13 root root    0 Sep  8 02:45 sys
drwxrwxrwt   4 root root 4096 Jul 27 18:25 tmp
drwxr-xr-x  46 root root 4096 Jul 27 18:25 usr
drwxr-xr-x  30 root root 4096 Jul 27 18:25 var
```

容器

使用Docker运行一个系统

- docker run -ti ubuntu:14.04

```
runqing@List-Cat:~$ docker run --rm -ti ubuntu:14.04
root@20eb8f36ce0f:/# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:ac:11:00:02
          inet addr:172.17.0.2  Bcast:0.0.0.0  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:20 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3723 (3.7 KB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

运行这个Ubuntu 14.04只需要下载180MB的镜像

docker

应用？系统？

Linux发行版

桌面系统（xface）

包管理工具（apt）

…

配置文件

应用

可执行文件

依赖库

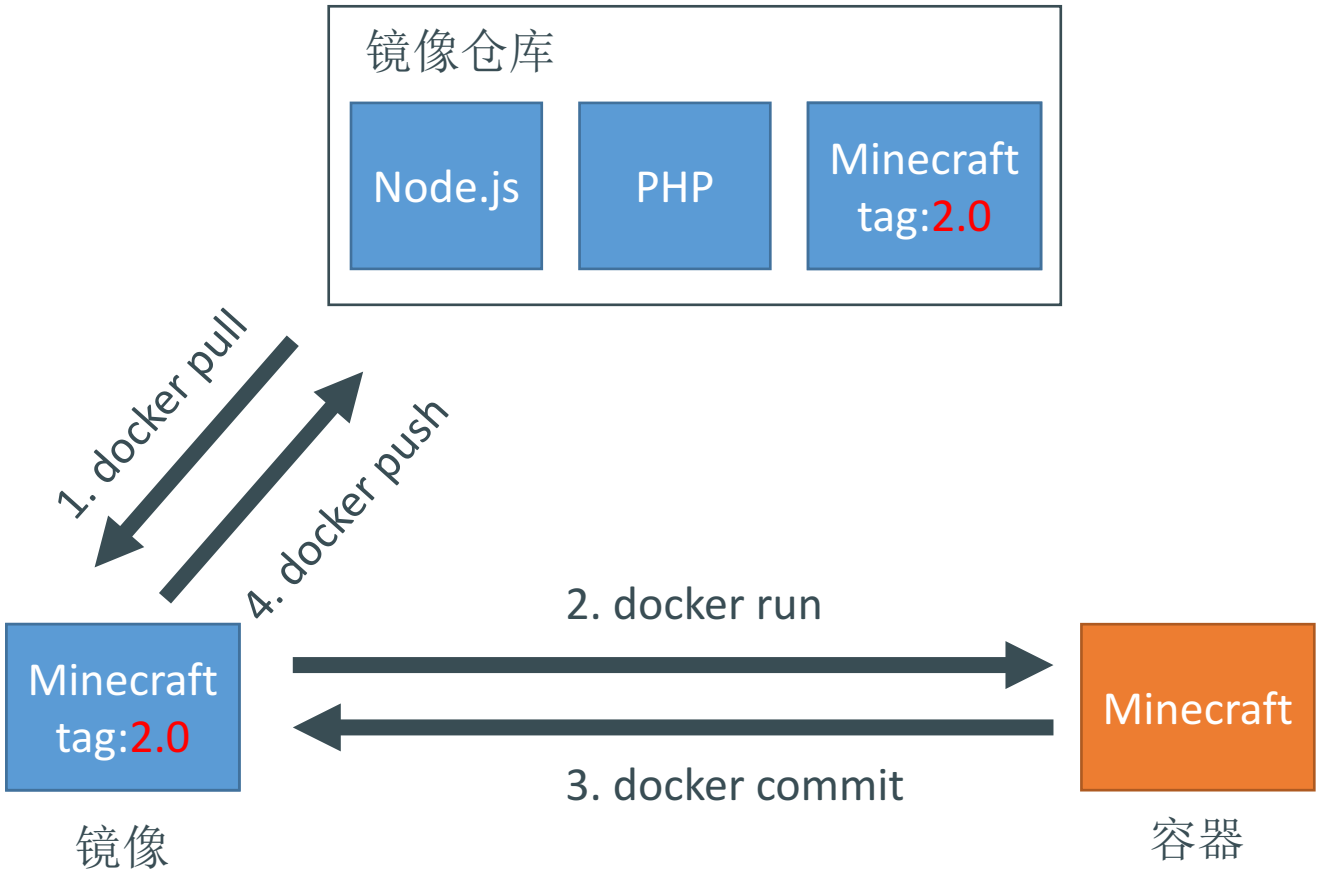
…

配置文件

基础概念

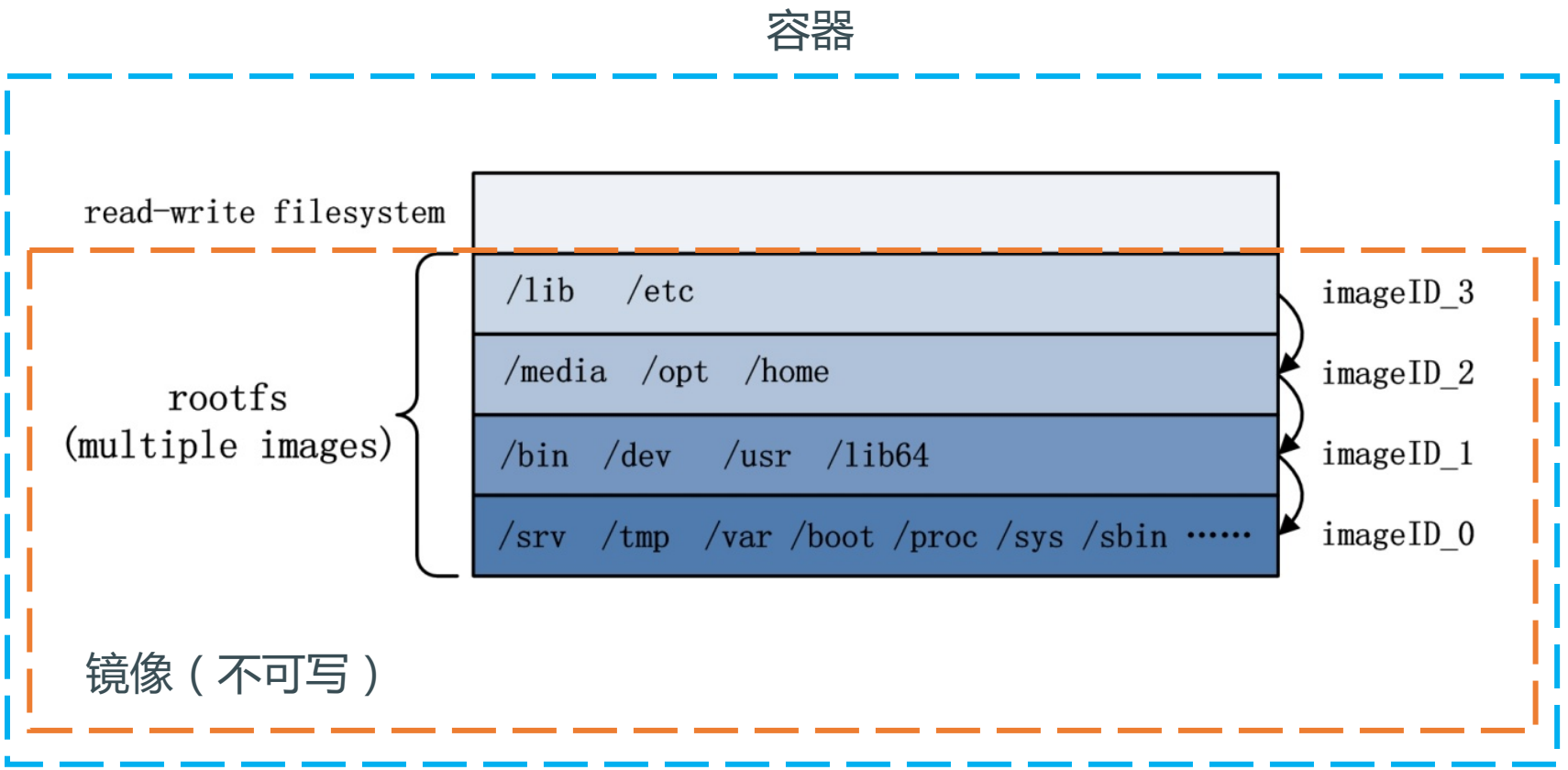
- 镜像（Image）：Docker **镜像**是一个特殊的文件系统，包含软件运行所需要的所有依赖和配置。
- 容器（Container）：容器是**镜像**运行时的实体，如同**类**和**实例**的关系。
- 镜像仓库（Registry）：一个存储和分发**镜像**的服务。

镜像、容器、仓库的关系



docker

镜像和容器的结构

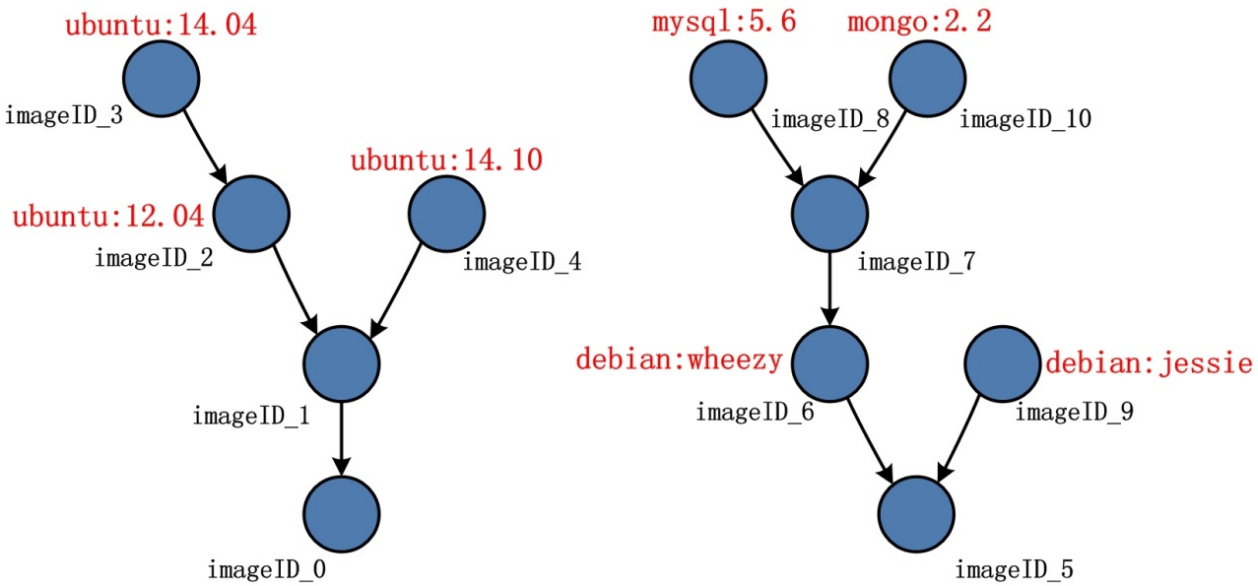


- 镜像是由多个镜像层组成
- 容器 = 镜像 + 可读写层

[1] 10张图带你深入理解Docker容器和镜像

镜像结构

多个镜像基于相同的镜像层

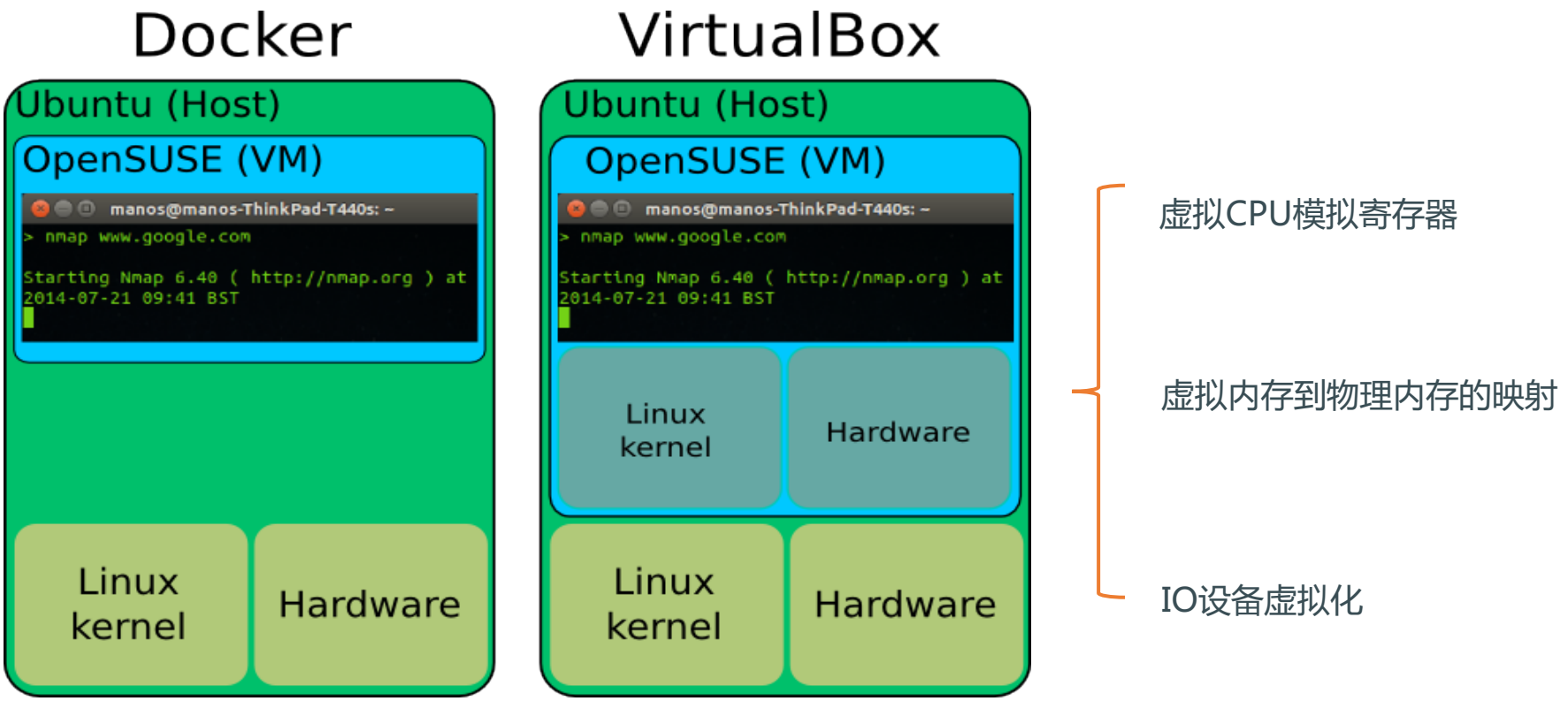


docker

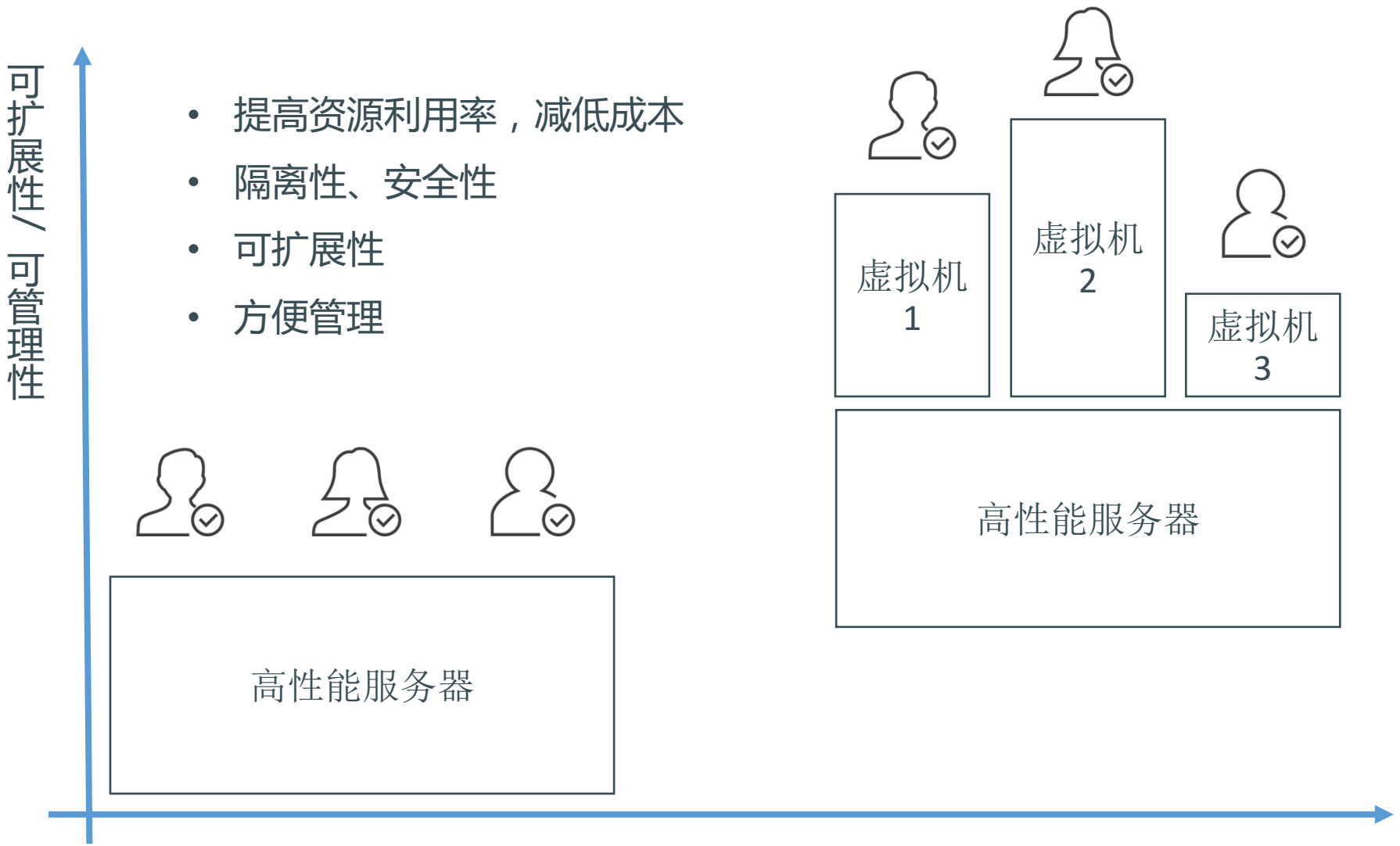
初步印象

实现原理上的区别

- 多个Docker容器共享一个内核
- 每个虚拟机拥有独立的操作系统（内核）



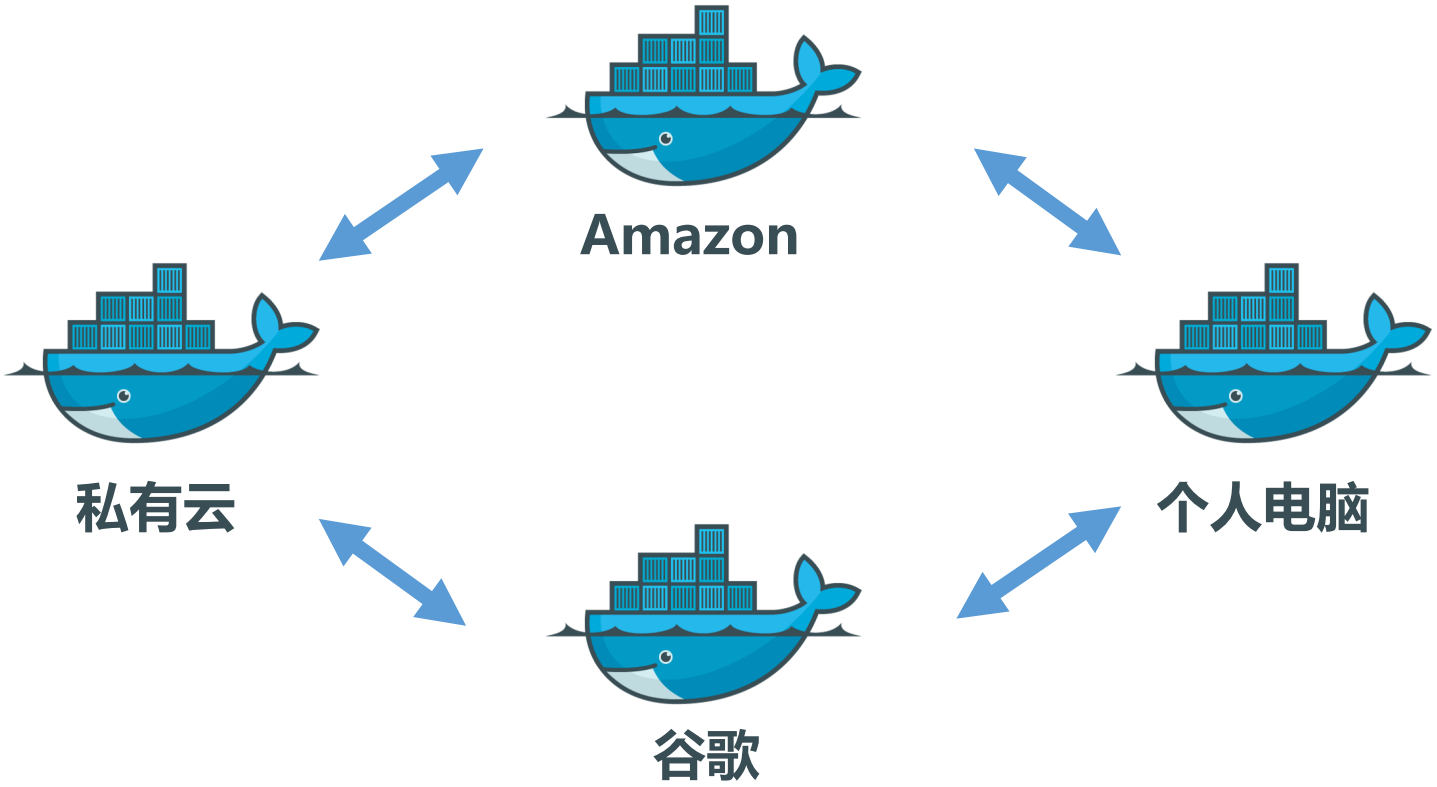
思想上的不同—面向基础设施



docker

思想上的不同—面向应用

- 集装箱：提供了一个标准化的方式来发布软件



Build, Test, Ship and Run Anywhere

docker

Docker VS 虚拟机

特性	容器	虚拟机
启动	秒级	分钟级
硬盘使用	一般为 MB	一般为 GB
性能	接近原生	弱于原生
系统支持量（服务器）	上百（千）个容器	几十个
硬件接口	直接使用	模拟硬件
安全性（隔离性）	低	高

普通用户角度

- 命令行操作，感到不舒服
- Linux上的Docker不能安装Windows系统，Windows上的Docker也不能安装Linux系统
- 默认的容器没有图形界面

开发者

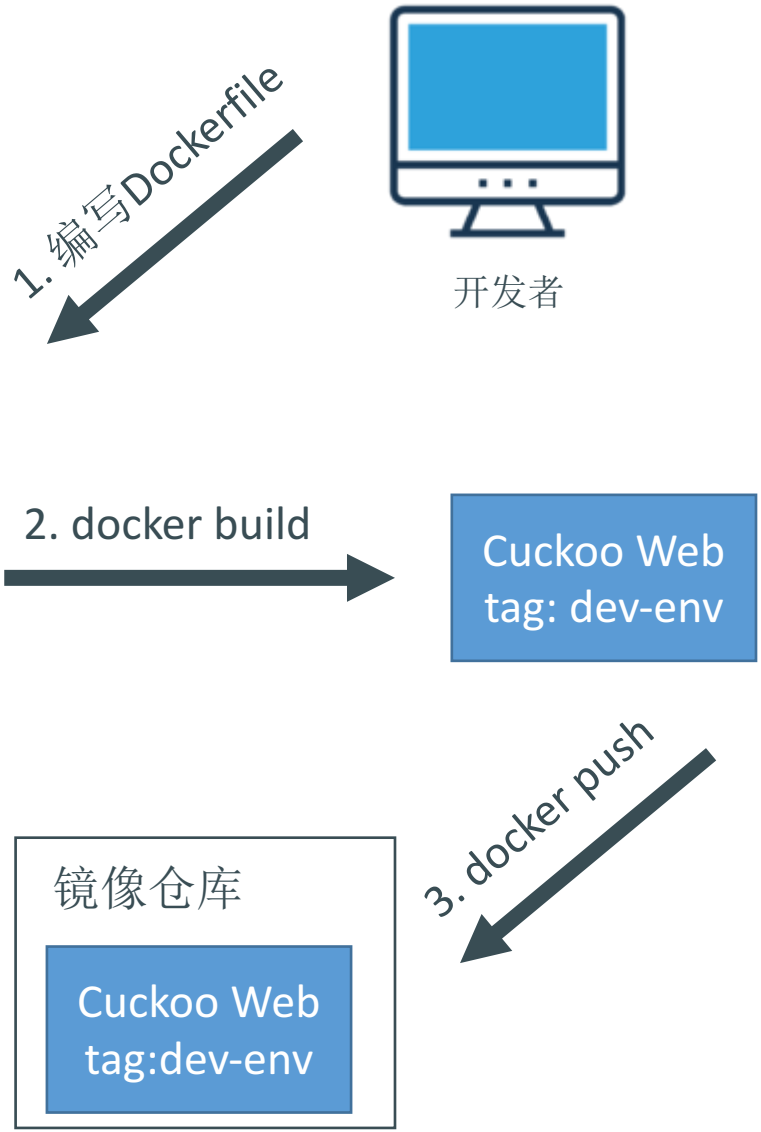
- 统一的开发环境！
- 减少“代码在我机器上就能跑”的问题
- 减少构建和配置开发环境的时间

全局角度：开发、测试、部署

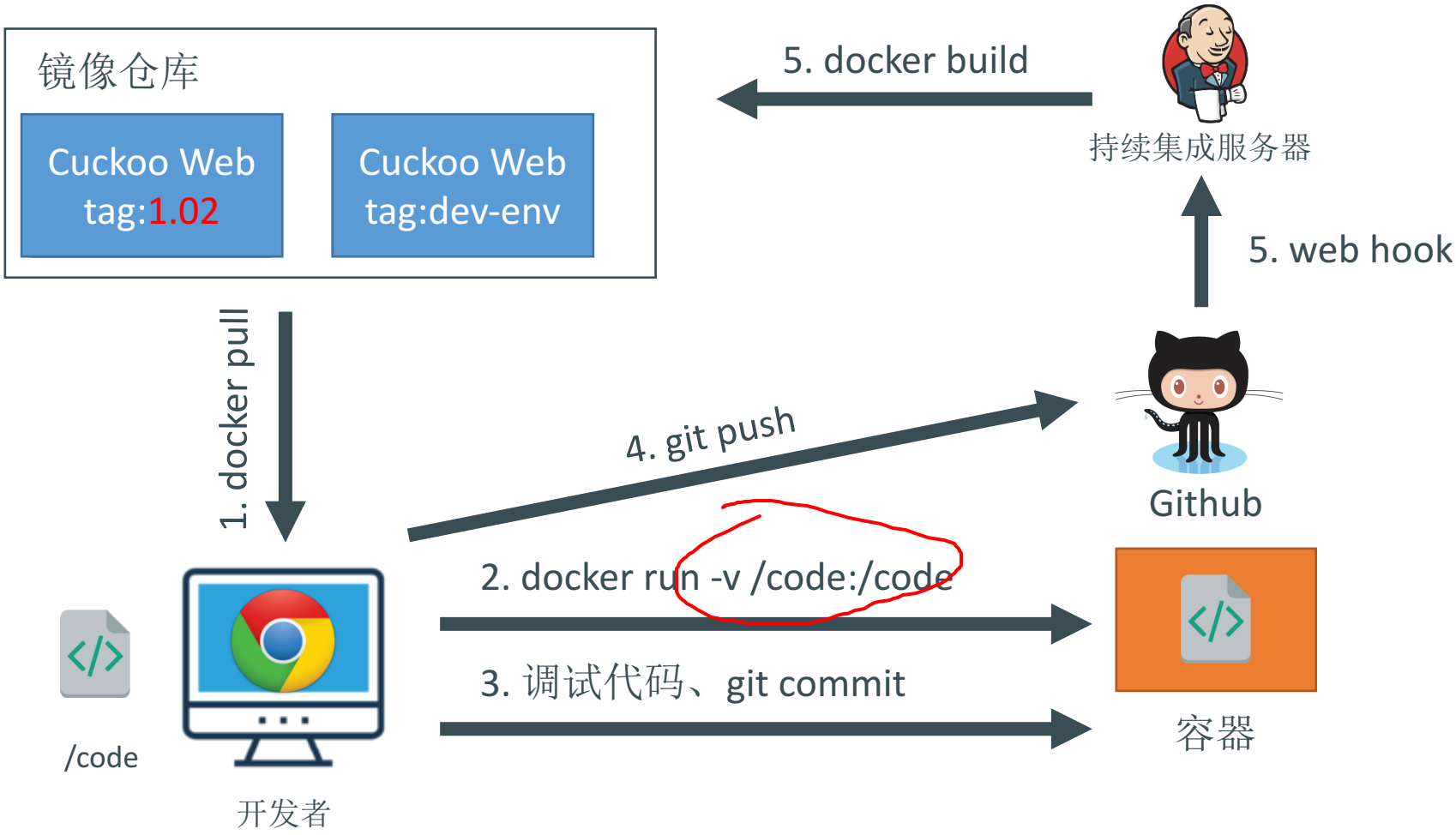
- 统一开发、测试、部署的环境！
- 测试发现的bug能够很好的重现！
- 不需要如何构建的readme，这些操作都被封装在Docker镜像中
- 持续集成、部署

案例—开发者视角

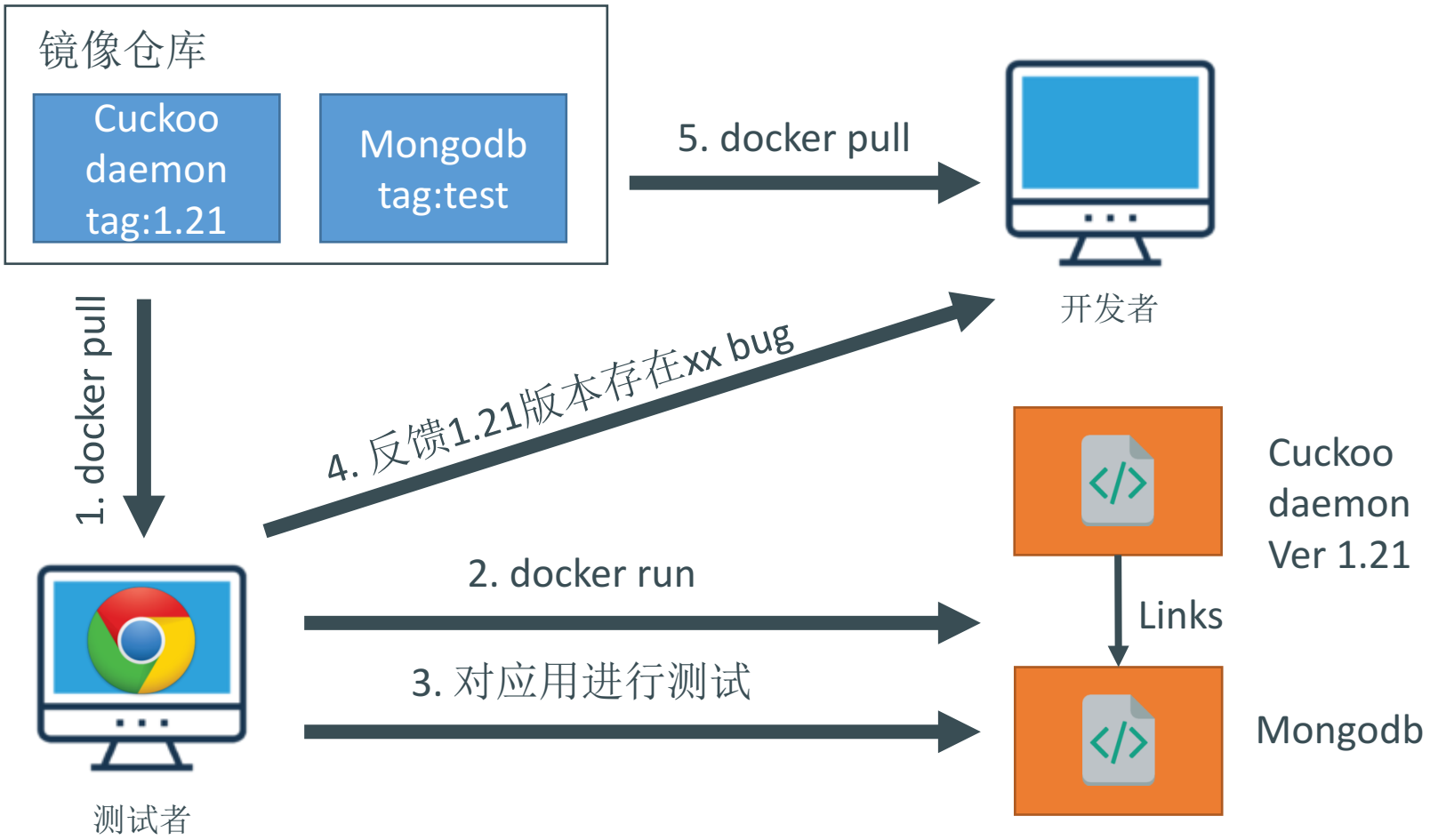
```
FROM ubuntu:14.04
LABLE maintainer xxx@gmail.com

# install dependences
RUN apt-get install python python-
pip python-dev libffi-dev libssl-dev
....

# add user
RUN groupadd -r cuckoo &&
useradd -r -g cuckoo cuckoo
....

ENTRYPOINT ["/entrypoint.sh"]
CMD ["--help"]
```



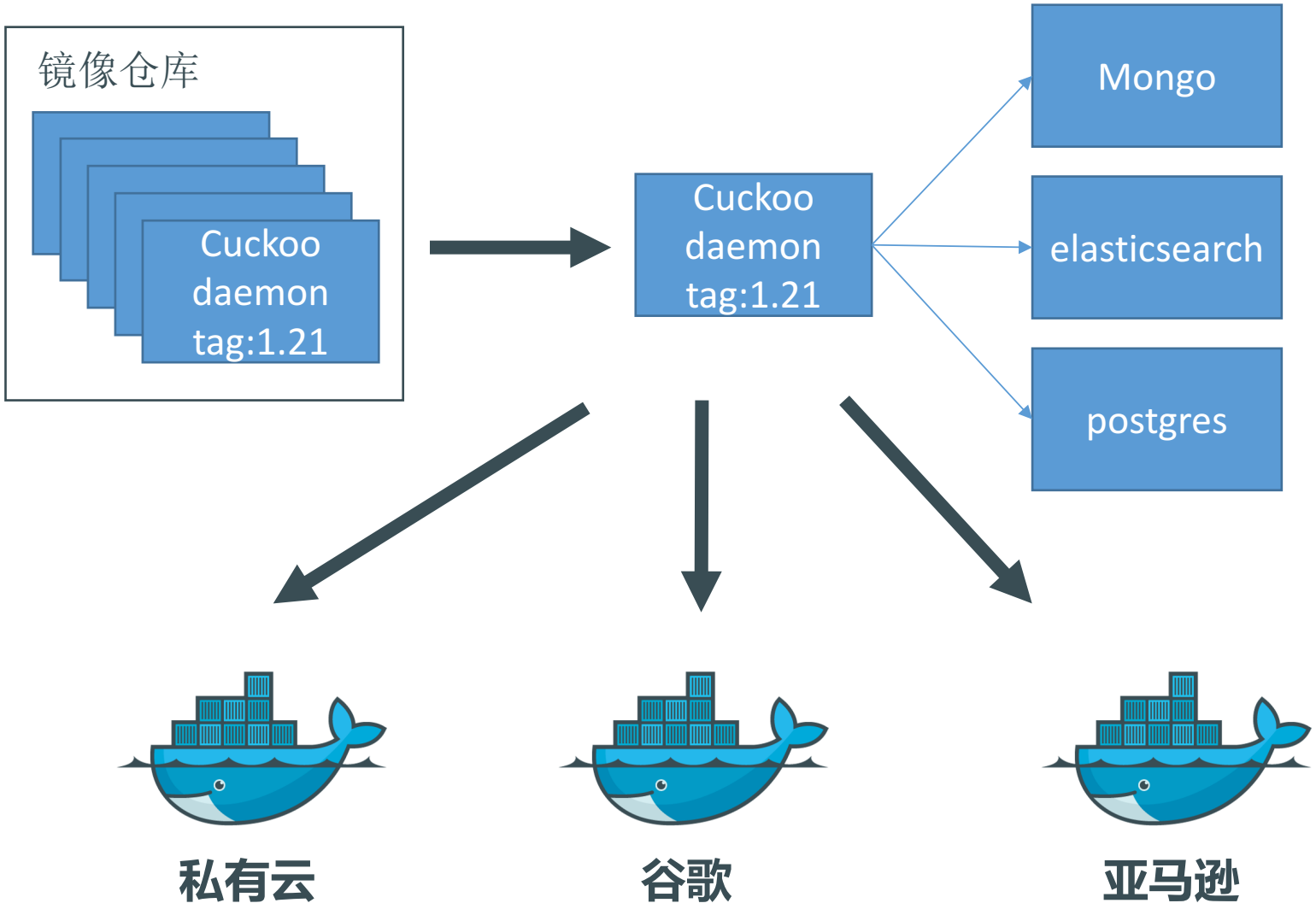
案例—开发者视角


镜像仓库
Cuckoo Web
tag:1.02
Cuckoo Web
tag:dev-env
5. docker build
持续集成服务器
5. web hook
1. docker pull
4. git push
Github
2. docker run -v /code:/code
3. 调试代码、git commit
/code
开发者
容器

案例—测试者视角


镜像仓库
Cuckoo daemon tag:1.21
Mongodb tag:test
5. docker pull
开发者
1. docker pull
4. 反馈1.21版本存在xx bug
Cuckoo daemon Ver 1.21
2. docker run
Links
3. 对应用进行测试
Mongodb
测试者

案例—运维视角



docker

Docker不是万能的

- 虚拟机一样能够统一开发环境

  Vagrant：用于快速构建虚拟开发环境的软件

```
$ vagrant init hashicorp/precise64

$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider.
==> default: Importing base box 'hashicorp/precise64'...
==> default: Forwarding ports...
default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Waiting for machine to boot...

$ vagrant ssh
vagrant@precise64:~$ _
```

- ✓ 镜像
- ✓ Vagrantfile自动配置镜像
- ✓ 镜像仓库[1]

[1] http://www.vagrantbox.es/

Docker不是万能的

- 如果把Docker当虚拟机来用，优势不大
- docker真正的优势：
  - 容器轻量，资源利用率更好（包括cpu、内存、硬盘空间等）
  - 统一的镜像分发，标准化了发布流程
  - 能够运行在任何安装了Docker的机器上，解决了底层基础环境的异构问题。
  - 服务器虚拟化解决的核心问题是资源调配，而容器解决的核心问题是应用开发、测试和部署[1]

[1] https://www.zhihu.com/question/48174633/answer/218092411

FAQ

Q : Linux上的Docker可以装Windows吗？

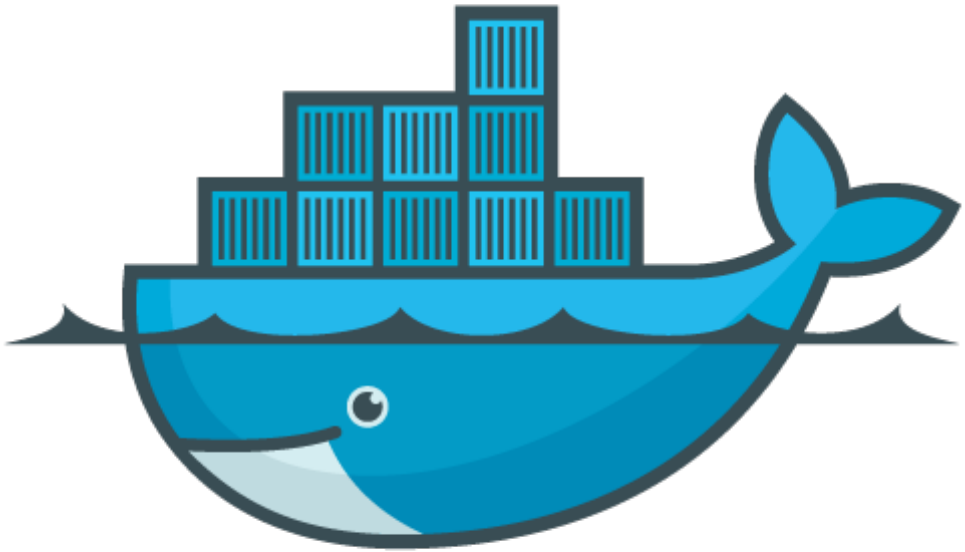
A : 不可以，因为linux和windows内核不相同。

Q : Docker可以运行有GUI的应用？

A : 可以。通过x windows转发。


不要跟我我说什么
阅读Readme
到Google上搜索
在Stackoverflow上提问
什么虚拟机，什么
脚本，什么添加参数
你们真是
不嫌麻烦
老夫搭环境就是
一把梭！
Python，Java，Node.js
统统一键包！
拿到Docker镜像就docker run，
it works就git commit

Thanks



docker

www.docker.io