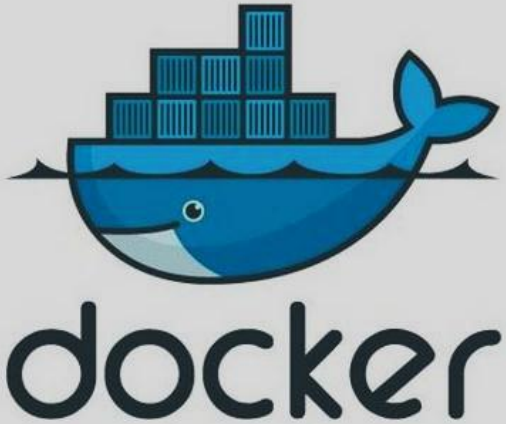
# 同程旅游5000个Docker实例上的CI实践

王晓波

CNUTCon2016 全球容器技术大会

主办方 Geekbang. 极客邦科技 InfoQ

# 目录/议程




- 同程旅游的Docker使用状况
- 在使用Docker前后CI/CD的挑战
- 基于Docker同程旅游的CI/CD做什么
- 对DEVOPS的实践
- 我们的一些经验

# 我们为什么要用Docker

- Linux物理机起步较晚，运维团队人手不够
- 爆发式增长，多应用部署导致环境混乱
- 各个业务应用系统抢占资源
- 物理机利用率不平均，资源利用不充分
- 无法快速扩容

# 我们的Docker使用现状

- Docker容器5000个，高峰扩容容器8000 个
- 容器应用包含：web应用，数据库，SOA微服务，消息队列...
- cpu利用率由20%提升为80%
- 原传统运维没有了
- Docker应用600个
- 塞入容器的还有：Mongodb， Redis，Mysql

# 我们从资源隔离层面做到了

- 物理机利用率的提升，合理的编排应用，让各物理机资源使用更 平均，减少买服务器的花费
- 各应用间资源隔离，避免环境和资源的冲突，提升安全性
- 爆发式流量进入，快速扩容和迁移
- 应用迁移——减少买服务器的花费
- 解放运维工作，更多的自动化，更精细化的监控和报警

# Docker是一个好选择

# 还有其它的问题是不是也可以

- 测试环境与生产环境的问题
- 持续部署时众多应用部署标准的问题
- 扩容时的环境一至性的问题
- 部署构建速度慢的问题
- 更快速的回滚问题
- 运维与开发的矛盾问题
- DEVOPS的实践

# 持续集成，持续交付，持续部署

- 我们面对是一个什么样的需求
- 怎么快速无技术难度的快速编程
  - 唯一不变的是需求
  - 唯一的要求是快

# 他们是什么

**持续集成**

一种软件开发实践，即团队开发成员经常集成它们的工作，通常每个成员每天至少集成一次，也就意味着每天可能会发生多次集成。每次集成都通过自动化的构建（包括编译，发布，自动化测试)来验证，从而尽快地发现集成错误。

**持续交付**

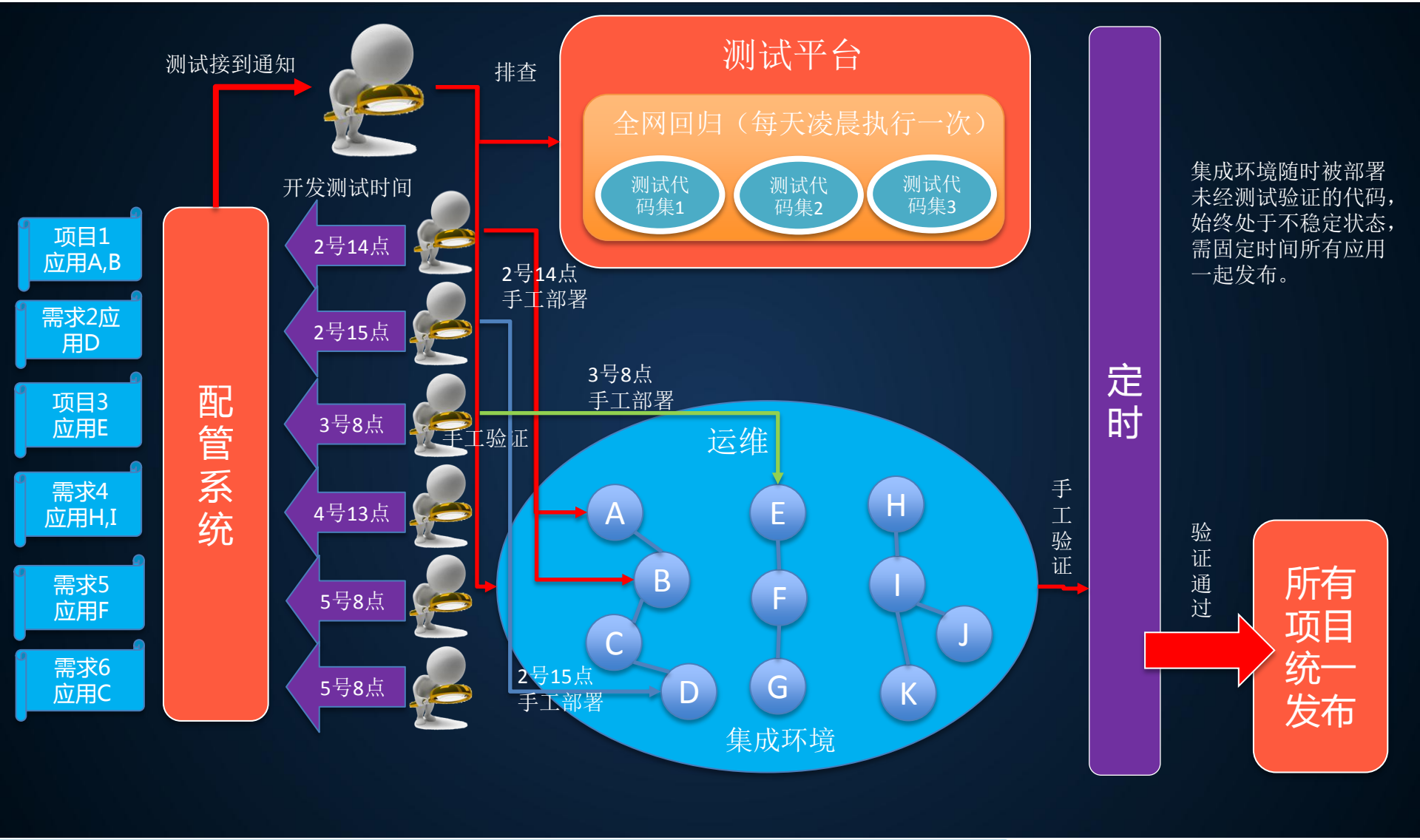
持续交付并不是指软件每一个改动都要尽快的部署到产品环境中。它指的是任何的修改都已证明可以在任何时候实施部署。

# 持续集成也是早于Docker

但是这个没有Docker为基础的CI坑真不少

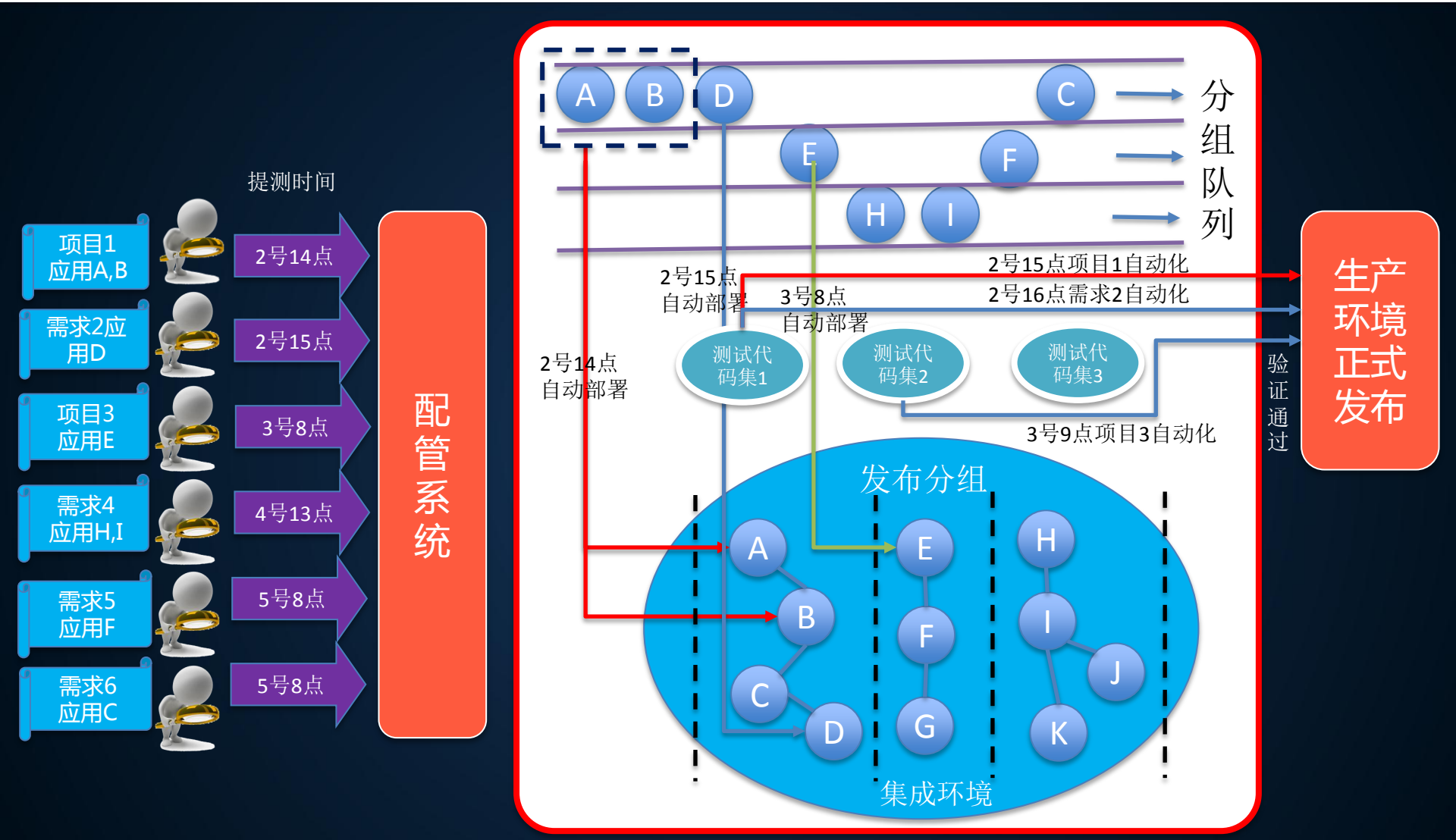
# 环境的坑

## 大集成环境 +固定发布时间(每周一次)



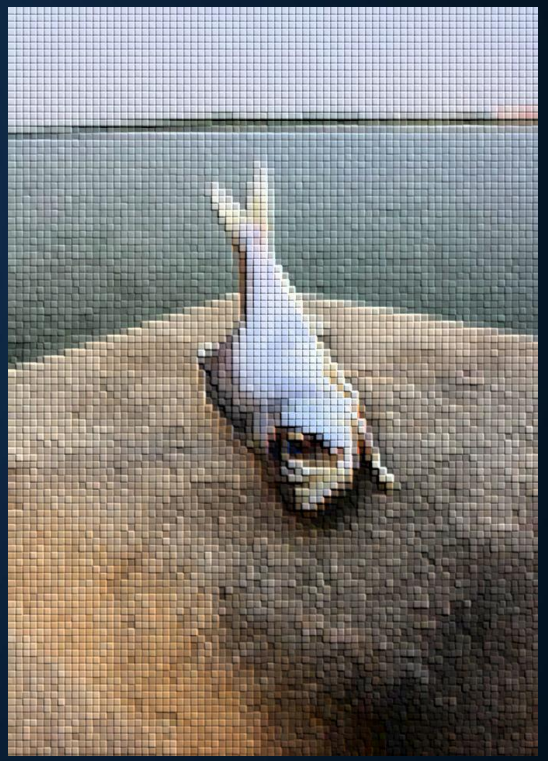
# 解决环境的问题

## 分组环境 + 随时发布



# 多个环境一致性又成新坑

- 开发环境与集成测试环境一致性问题
- 多个分组测试的集成测试环境一致性问题
- 测试环境与生产环境一致性问题



# 随时随地建立标准一致的环境

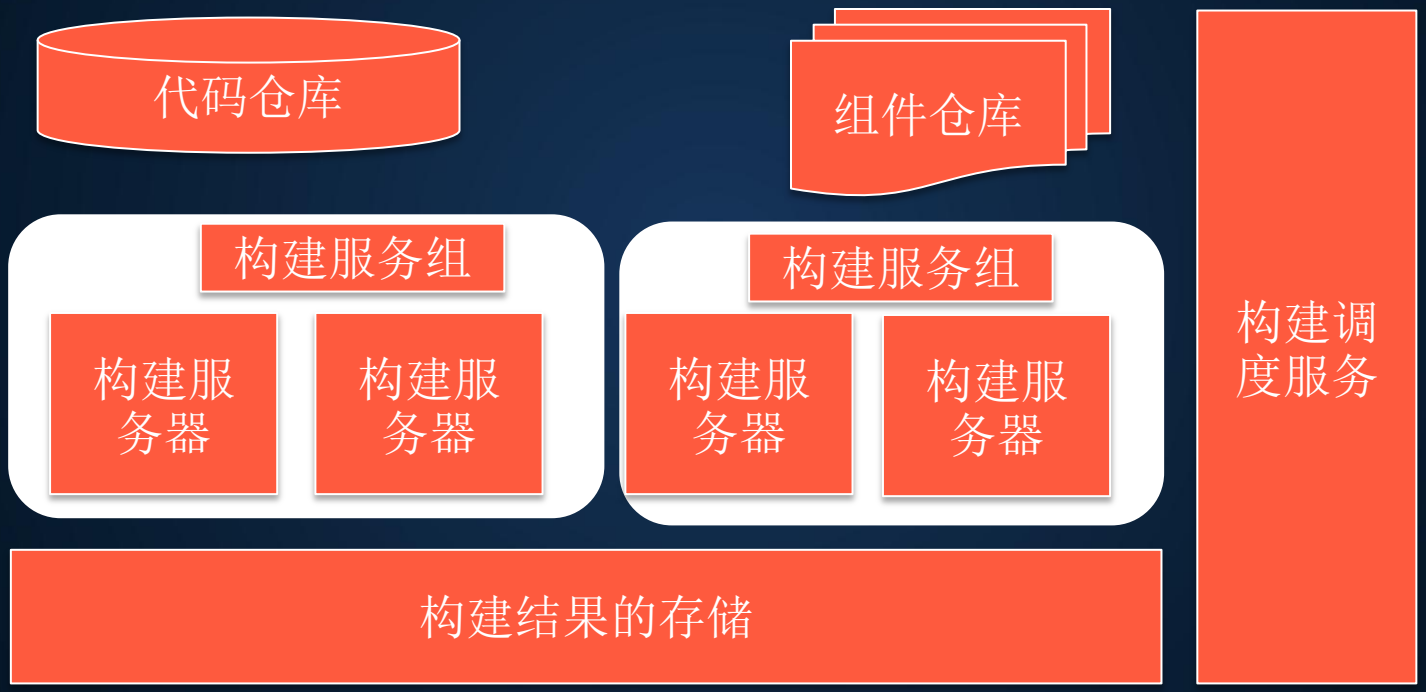
- 需要一个标准环境的模板池
- 每次建立环境时从标准池中拉出
- 有个良好的应用调用治理过程

# 代码构建之坑

- 构建慢，长长被吐槽
- 需要保存多少个版本来保证可回滚
- 代码依赖关系
- 多语言同一个平台构建

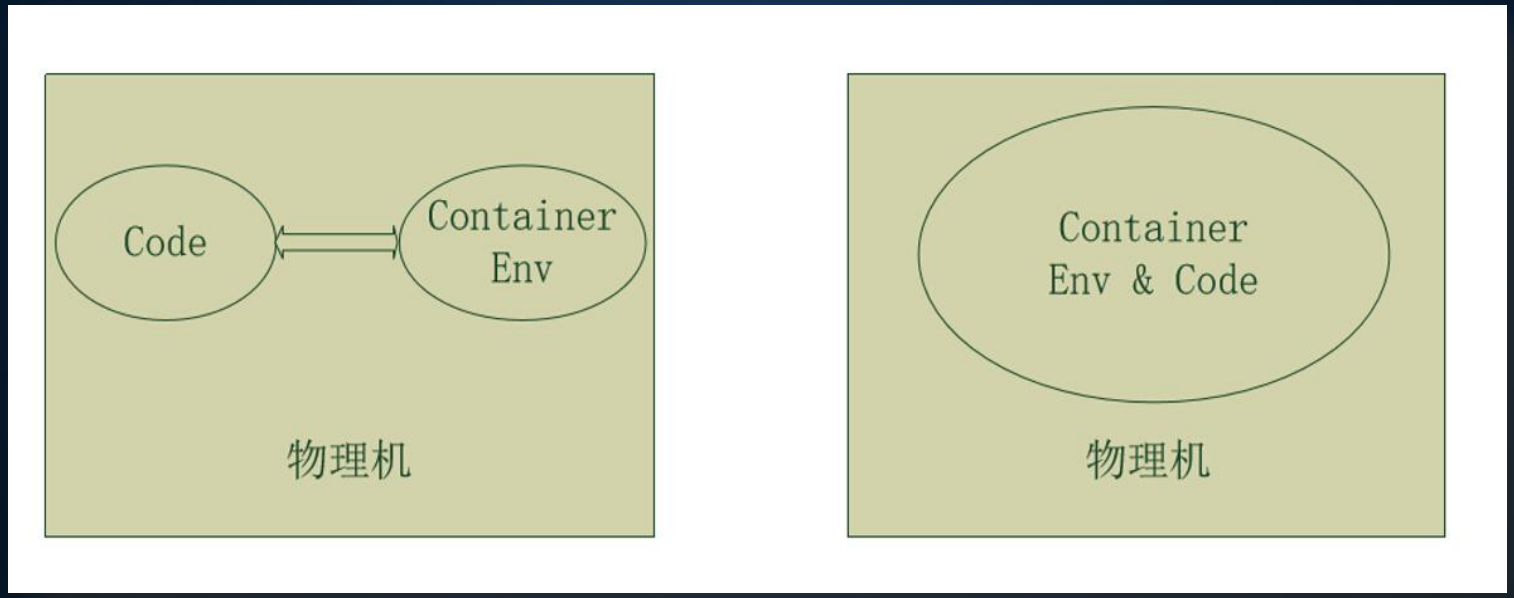

# 构建多快才是快

- 细细想想要不是有多快其实是顺



# 代码与容器

- Container 就是Container，它不是 虚拟机
- 代码在宿主机上，再挂载入镜像
- 代码打在镜像中，直接启动Container

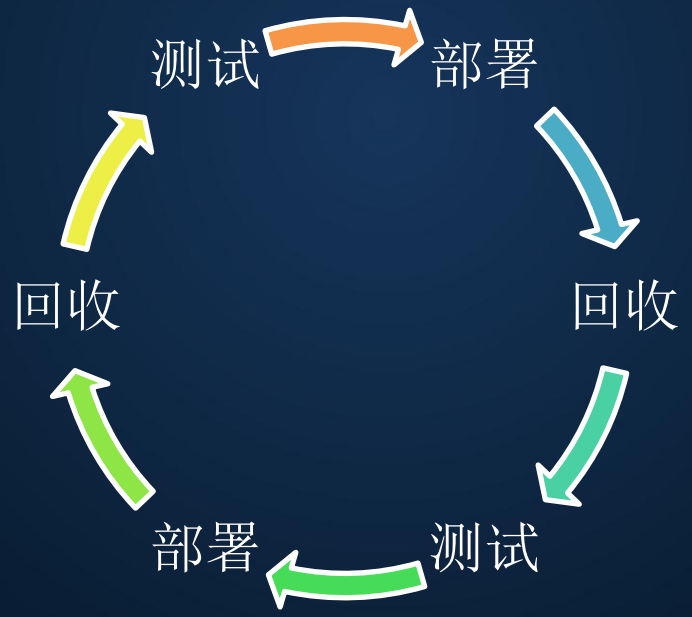


# 遵守Docker理念的发布

- 代码在Containr内每次发布制作镜像
- alpine镜像替换centos，200Mb　瘦身到5Mb
- Dockerfile命令优化，原来20层缩小为5层，构建速度快1倍
- 存储驱动从devicemapper改到overlayfs，构建速度快1倍
- Docker引擎升级到1.11
- 原来发布一个较大应用5分钟，现在40秒

# 容器与自动测试的结合

- 自动测试系统直接调用容器系统部署环境
- 可以要求测试要求快速部署
- 测试完成就可回收，供其它测试使用



# 扩容/排障等算不算持续集成/发布

## 有可能快速扩容并不快（因为错误变慢）

新加入的服务基础配置有误
新加入的服务用的代码是老的
新加一个服务发布的时间太长

## 排除故障大于一切

运维的各种黑科技（最终多不是）
怎么放弃被这些黑科技的结果
故障未除被搞出新故障

# 扩容/排障系统/调度/编排与CI/CD整合

DEVOPS门户

监控平台 | 调度管理平台 | 调用链系统

开发

代码仓库

CI/CD系统

容器管理系统

镜像仓库

配置中心

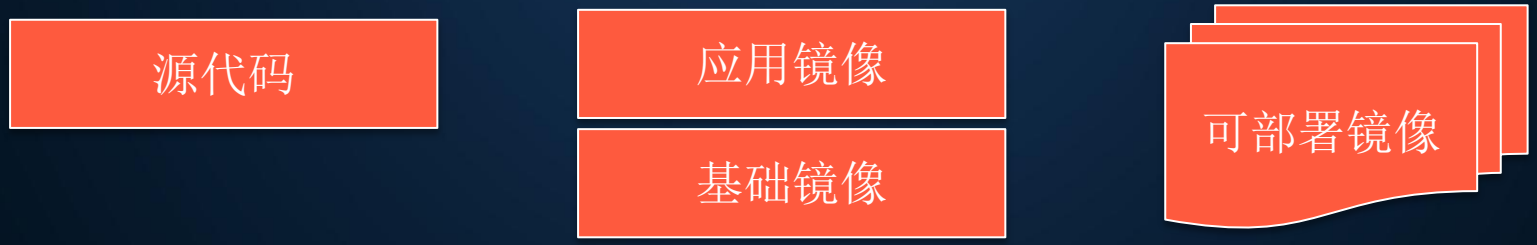
Robot Docker

Robot Docker

Robot Docker

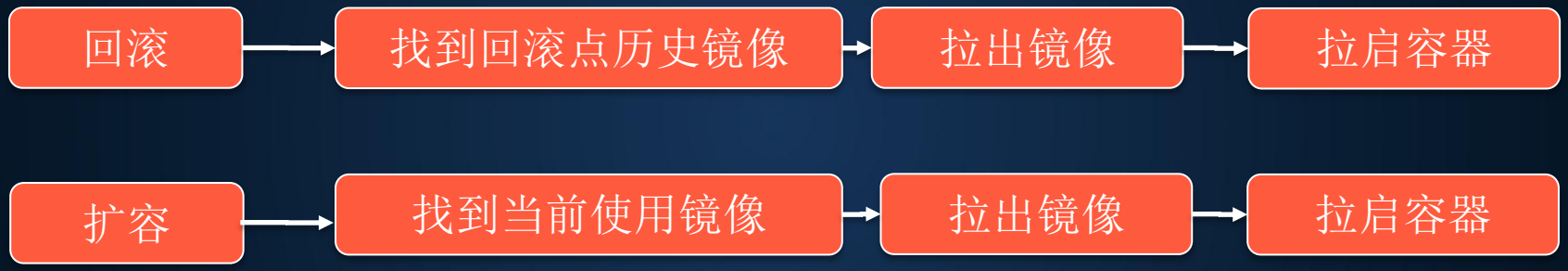
# 镜像的管理

- 基础镜像池的建设
- 基础镜像之上再构建应用镜像
- 应用镜像每次发布时重新构建
- 应用镜像多版本存储
- 一次构建镜像各处使用



# 使用镜像的回滚，扩容等

- 各应用的回滚，扩容全部居于应用的镜像来完成



# 运维与开发

- 一对欢喜冤家
- 我们看一段对话

开发:我们的应用为啥不能访问了?
运维:刚刚服务器内存坏了,服务器自动重启了啊
开发:为什么我们的应用延迟这么大?
运维:大哥,你老的代码跑一下3秒，你慢还是我的问题?
测试:我测试的时候这功能是对啊，生产怎么就错了呢
运维:这个可能环境不一样
开发:你们俩搞搞清楚吧
运维:好吧全是我的错，我就是专业背锅的
开发:下次我来做环境
运维:大哥,你以为这个是写几条命令啊

# DEVOPS可不是让开发去干运维

## 让开发做运维可不可行？ 1

可行但不靠谱（开发的水平也不平均）
开发与运维的思路不一样
开发做运维问题可能更大

## 2 让运维做开发？

要做开发，但不是开发业务代码
开发技术树是层底开发
运维要有架构的思想

## 那DEVOPS是什么？ 3

开发参与运维，运维参与开发，
但一定是基于高效的工具

# 运维基于容器的转型

- 是容器系统的开发者
- 运维系统的开发者
- 故障的产品经理

# 我们的一个实例


天梯 - SkyLadder v1.0.6
项目
个人
工作台
运维
系统设置
权限设置
帮助
项目查询
tcbase.cilab - 自动发布演
构建历史
部署历史
发布申请历史
正式部署
回滚
构建版本:#375223 v2.0.3.3
发布说明:测试
申请时间:2016-08-18 17:30:29
正式环境
服务器组KEY：product
高级
服务器名
IP
TCWEB065
172.16.2.56
TCWEB066
172.16.2.57
WinAgent版本
0.9.66.0
0.9.66.0
Skyladder
回滚部署
指定回滚版本
部署标识
构建标识-版本号
部署时间
回滚
419791
375083-v2.0.9.9
2016-08-18 16:25:56
历史版本
418658
370651-v2.0.8.9
2016-08-18 10:24:19
411770
368592-v2.0.8.9
2016-08-15 13:18:25
409200
366571-v2.0.8.9
2016-08-12 10:45:39
407681
364849-v2.0.8.9
2016-08-11 16:07:46
405963
364081-v2.0.8.9
2016-08-10 21:06:37
405933
364048-v2.0.8.9
2016-08-10 20:56:49
404243
361404-v2.0.8.9
2016-08-10 13:41:04
385532
346670-v2.0.8.9
2016-07-29 13:36:25
383307
344661-v2.0.8.9
2016-07-28 14:37:10
关闭
首页
前一页
下一页
尾页

# THANKS!