

# 如何用容器提升

开发工作效率

# 关于我

- 刘彬
- OneOaaS,研发部
  - 负责产品研发
- 关注云计算与大数据的运维
- Github: <https://github.com/heidsoft>



# 议题:

1. 软件开发面临的一些问题
2. 容器在开发中能做什么
3. 我在开发工作中如何使用容器

# 软件开发面临的一些问题

# 追溯历史,“软件危机”

“软件危机”是指软件开发的**现有方法**中强调需要**如何应对变化**的一系列问题。

该术语起源于**20世纪60年代**，大概是1968年在北大西洋公约组织（**NATO**）的软件会议的一篇文章“**Software Engineering**”中提出来的。

# “软件危机”的三个阶段

60年代~70年代



80年代~90年代



2005年至今

特点：

使用机器语言  
汇编语言在特定  
机器上进行软件设计开发  
计算能力弱

难以维护，难以移植

特点：

软件变得更加复杂，规模更加庞大  
《人月神话》中提及，  
IBM公司开发的OS/360系统共有  
4000多个模块，约100万条指令，  
投入5000人年，耗资数亿美元，  
结果还是延期交付。  
在交付使用后的系统中  
仍发现大量（2000个以上）的错误。

特点：

以aws为代表，从05年开始  
逐步步入云计算时代、大数据时代  
  
管理运维庞大的计算资源  
以及如何处理好海量数据

# 总结“软件危机”下的问题

## 典型问题代表

- 不可靠
- 交付延迟
- 修改成本过高
- 不可维护
- 执行水平不足
- 超出预算成本



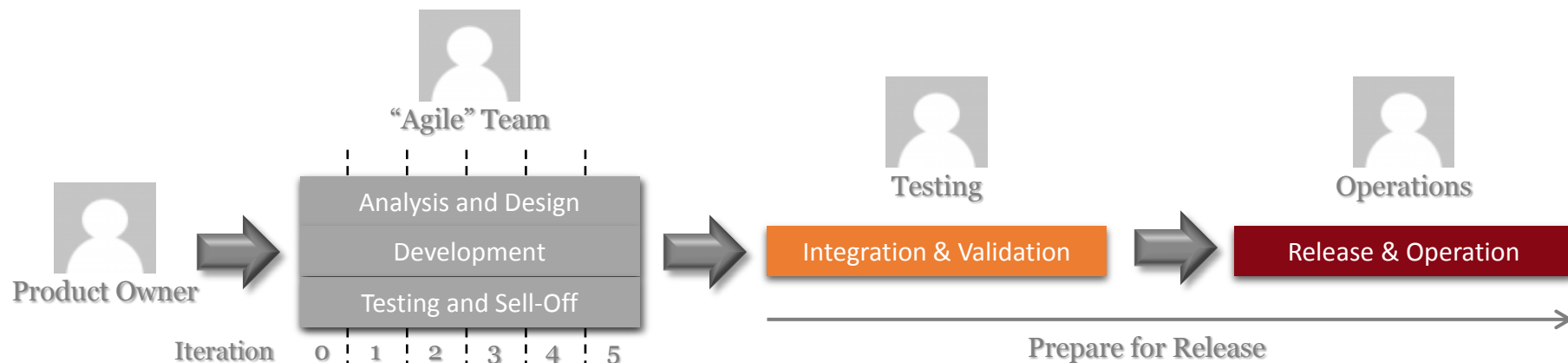
# 造成“软件危机”的主要原因

因为计算机的计算能力正在**呈指数级地增长**！说的简单些：在没有计算机的时候，**编程根本就不是一个问题**；当一些计算能力较弱的计算机出现时，**编程成了一个中等难度的问题**，而现在，我们拥有了计算能力超绝的计算机，**编程就变为了一个同样复杂的问题**。

– Edsger Dijkstra, 1972年图灵奖获奖感言



# 当前普遍状况



- 一个版本的发布需要较长时间
- 在集成过程中发现的程序问题通常较难处理和解决
- 各阶段相互依赖并需要等待时间
- 需要创建多分支来进行软件开发
- 不能与Agile原则一致

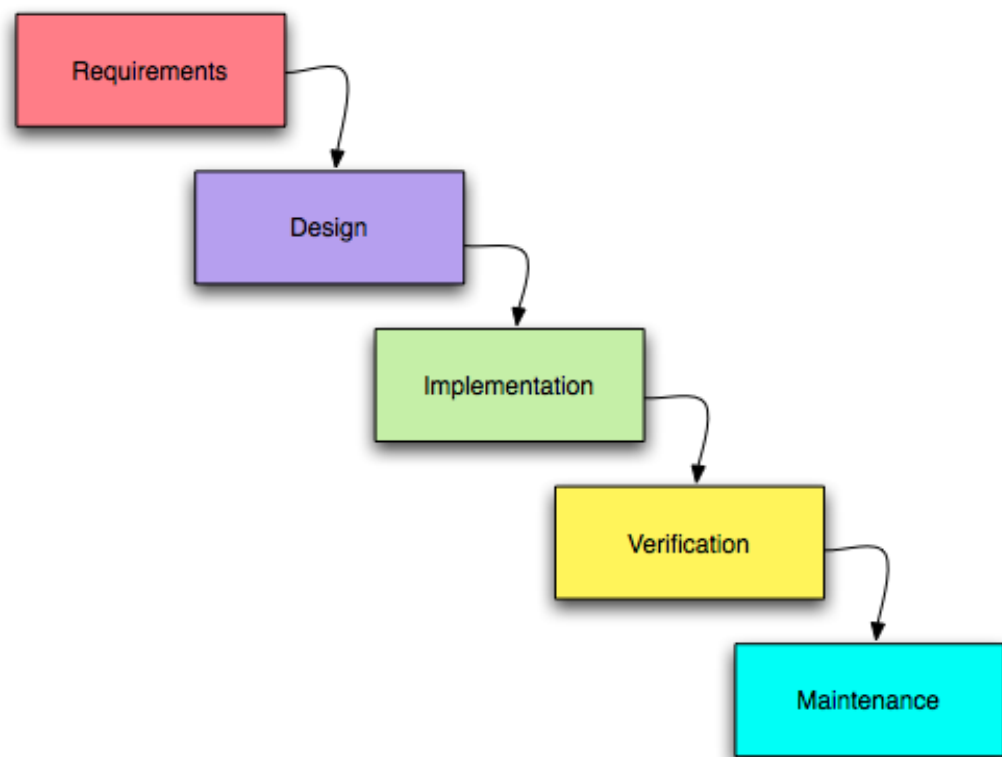
# 如何解决这些问题？

使用“模式”来解决其中一些问题，典型的软件开发模式有：

- 软件开发模式1 -瀑布模式
- 软件开发模式2 -迭代开发模式
- 软件开发模式3 -敏捷开发模式
- 软件开发模式4 -极限开发

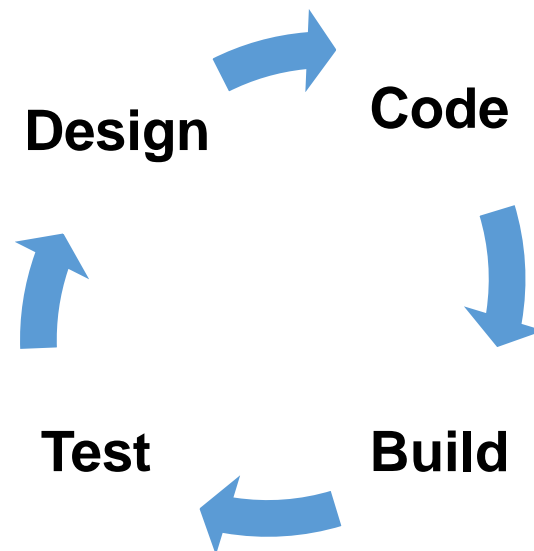
# 软件开发模式1-瀑布模式

- 瀑布模型式是最典型的预见性的方法，严格遵循预先计划的需求分析、设计、编码、集成、测试、维护的步骤顺序进行。
- 步骤成果作为衡量进度的方法，例如需求规格，设计文档，测试计划和代码审阅等等。



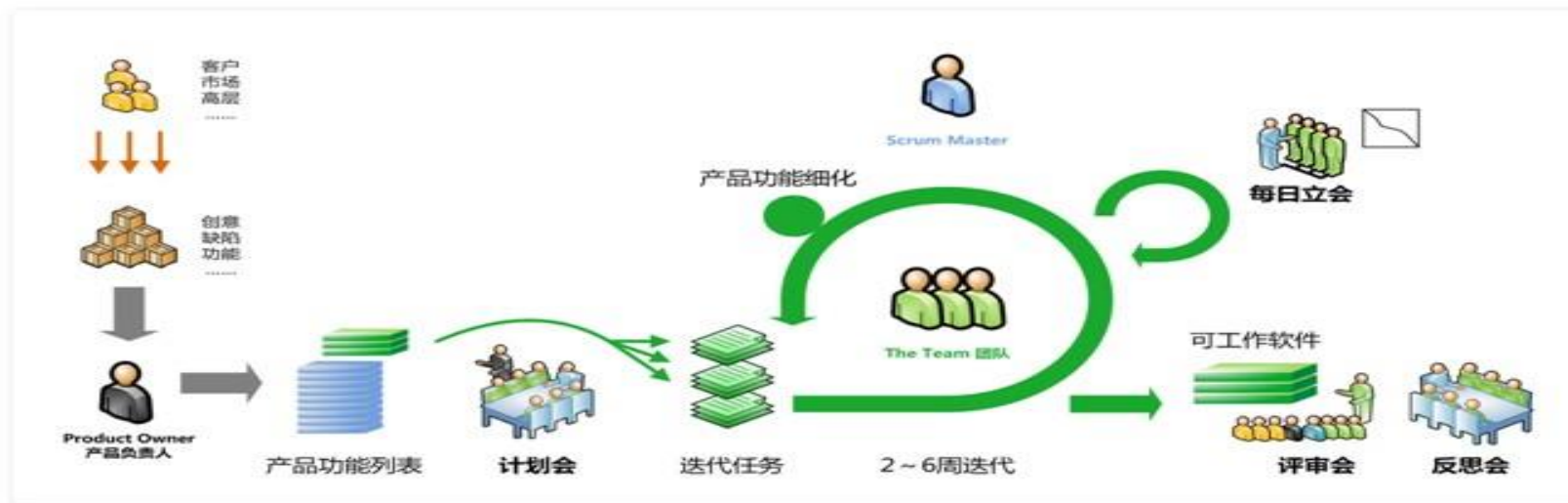
# 软件开发模式2 -迭代开发模式

- 迭代式开发也被称作迭代增量式开发或迭代进化式开发，是一种与传统的瀑布式开发相反的软件开发过程，它弥补了传统开发方式中的一些弱点，具有更高的成功率和生产率。
- 每次只设计和实现这个产品的一部分
- 逐步逐步完成的方法叫迭代开发
- 每次设计和实现一个阶段叫做一个迭代.



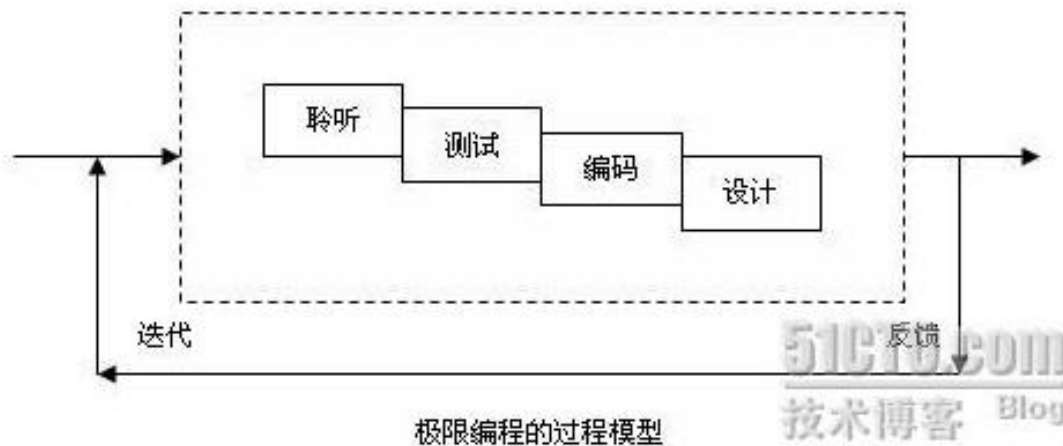
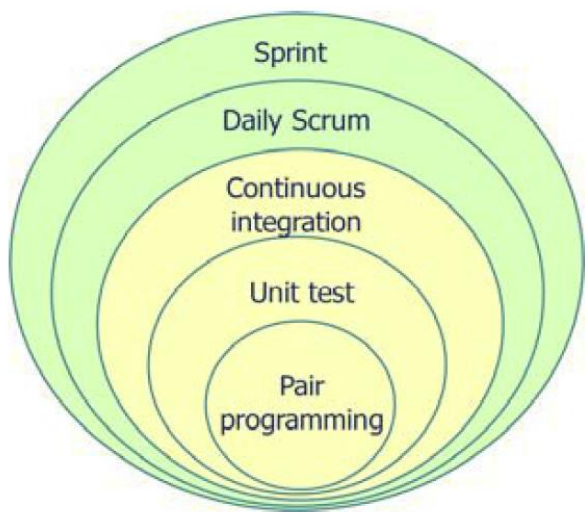
# 软件开发模式3 -敏捷开发模式

- 是一种应对快速变化的需求的一种软件开发能力。更强调程序员团队与业务专家之间的紧密协作、面对面的沟通（认为比书面的文档更有效）、频繁交付新的软件版本
- 而自我组织型的团队、能够很好地适应需求变化的代码编写和团队组织方法，也更注重软件开发中人的作用。
- 人和交互 重于过程和工具。
- 可以工作的软件 重于求全而完备的文档。
- 客户协作重于合同谈判。
- 随时应对变化重于循规蹈矩。



# 软件开发模式4 - 极限开发

- XP是一种近螺旋式的开发方法，它将复杂的开发过程分解为一个个相对比较简单的小周期；通过积极的交流、反馈以及其它一系列的方法，开发人员和客户可以非常清楚开发进度、变化、待解决的问题和潜在的困难等，并根据实际情况及时地调整开发过程。



如何**用好敏捷**开发模式

# 将敏捷延伸到持续交付



形成快速的质量反馈

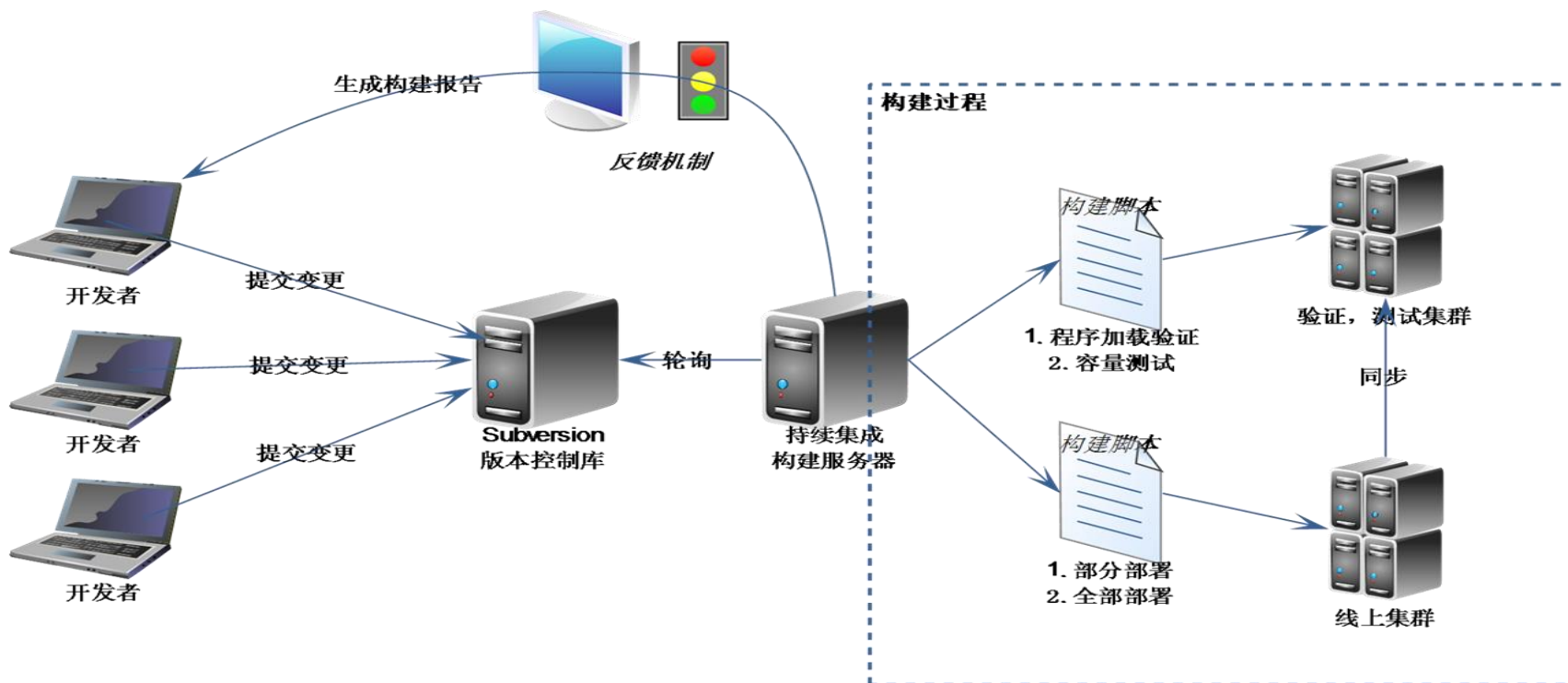
- Agile 需要快速的迭代过程
- CI 能够在每次检入时自动编译/单元测试/代码检测
- CT 能够自动的对构建的结果进行功能性测试
- CD 能够按需频繁地向生产环境发布

**REPEAT!**



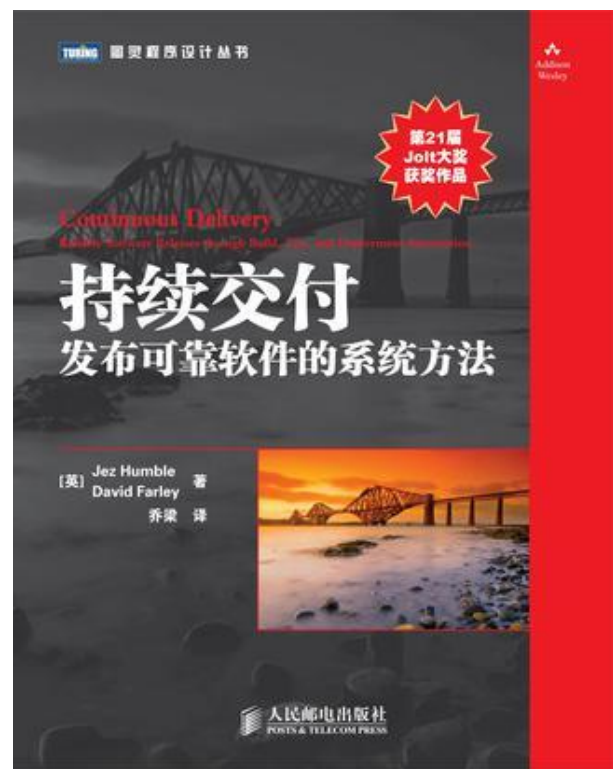
# 持续集成

随着软件项目复杂度的增加（多人协作开发），就会对集成和确保软件组件能够在一起工作提出了更多的要求-要早集成，常集成。**早集成，频繁的集成**帮助项目在早期发现项目风险和质量问题，如果到后期才发现这些问题，解决问题代价很大，很有可能导致项目延期或者项目失败。



# 持续交付

- 持续交付是一种方法论，涵盖
  - 持续集成Continuous integration
  - 自动化测试Automated testing
  - 在不同环境下可持续自动化部署Continuous Deployment
- 目标是改变软件交付方式，将其由开发人员的手工操作变成一种可靠、可预期、可视化的过程并在很大程度上实现了自动化的流程，而且它要具备易于理解与风险可量化的特点。
- 有可能在几分钟或几个小时内把一个想法/补丁快速精准地交付到生产环境/客户，而且同时还能提高交付软件的质量。

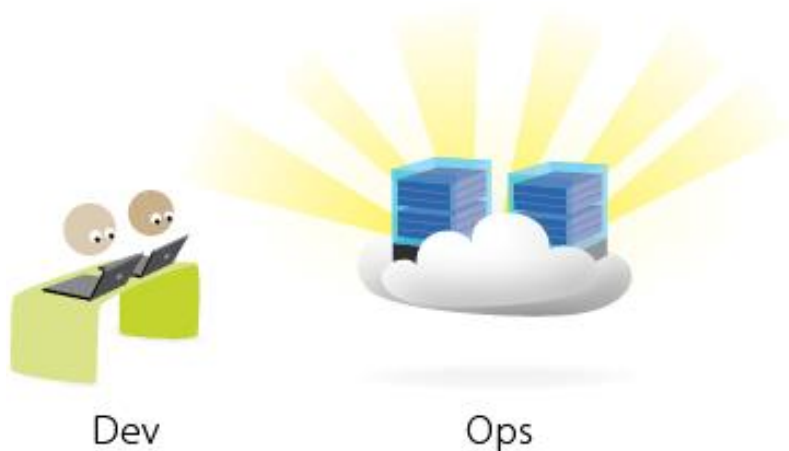


# 持续交付面临的挑战

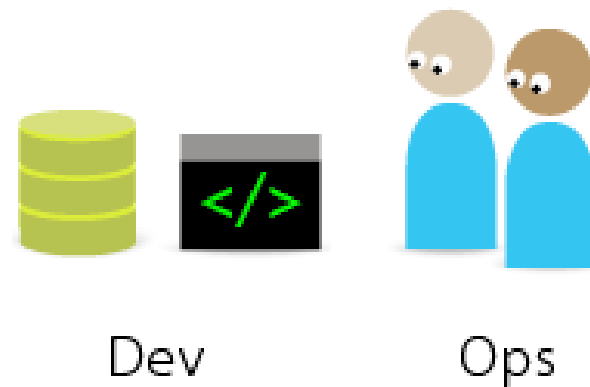
- 目前常见做法:
  - 完全手工实现软件发布或部署
  - 只能在软件开发完成后才开始发布或部署到生产环境中
  - 手工的软件开发环境的配置管理
- 技术方面挑战
  - 软件开发过程基于人工的从一个到另一个环境(dev, test, UAT, pre-production, production)迁移，耗时且出错率高
  - 较长的冲突解决过程
  - 缺乏足够的开发和测试的迭代循环
  - 由于大量的bug导致的发布延迟
  - 每次发布过程不可重复，大量重复工作
- 业务影响
  - Time to market周期长
  - 软件开发和测试成本高
  - 软件质量难以提高
  - 客户满意度低

# 软件交付的发展-Developer（Dev）和 Operator（Ops）

Dev 做 Ops的事情



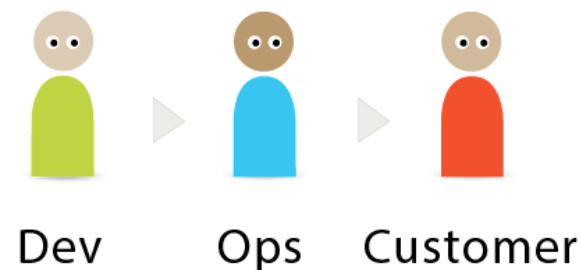
Ops 使用Dev的工具和流程



Dev和Ops之间的障碍

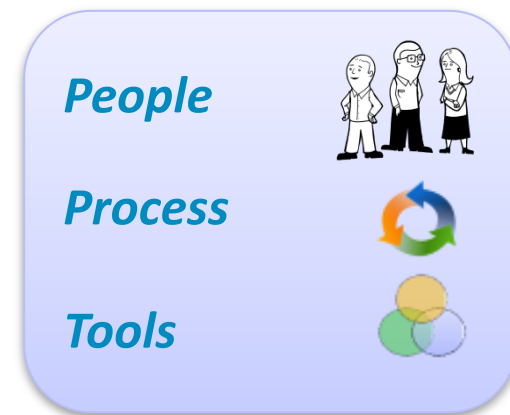


Dev+Ops=DevOps

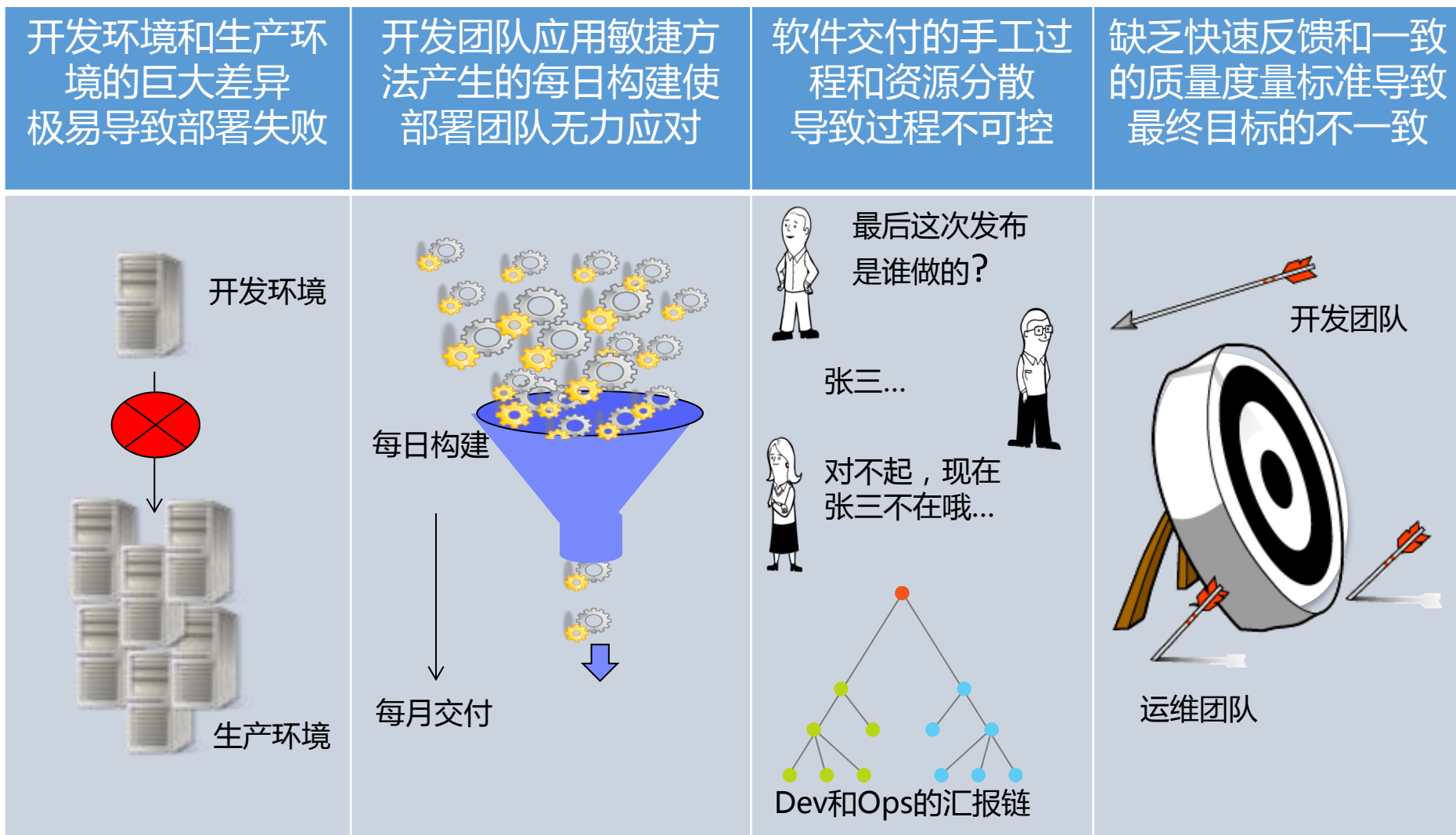


# DevOps的定义—DevOps，让持续交付成为可能

- 持续交付是指，以自动化或半自动化方式，将构建版本从一个环境提送（**promote**）到更接近实际生产的交付准备环境。
- DevOps于08年开始在欧洲开始流行，用于促进开发、技术运营部门之间的沟通、协作与整合。它的出现是由于软件行业日益清晰地认识到：为了按时交付软件产品和服务，开发和运营工作必须紧密合作。
- DevOps的四个原则：
  - 一开始就在类生产环境上开发和测试
  - 使用可重复、可靠的过程进行迭代化、高频率的部署
  - 持续的监控和验证软件运行的质量特征
  - 扩大软件交付反馈圈



# Dev和Ops之间的障碍是什么？



# DevOps的价值



## 加速软件交付

扩大协作，使之包括客户、业务条线和其他相关人，以便更好的消除组织管理壁垒。

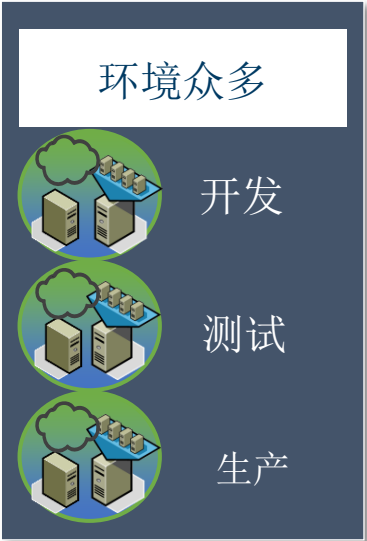
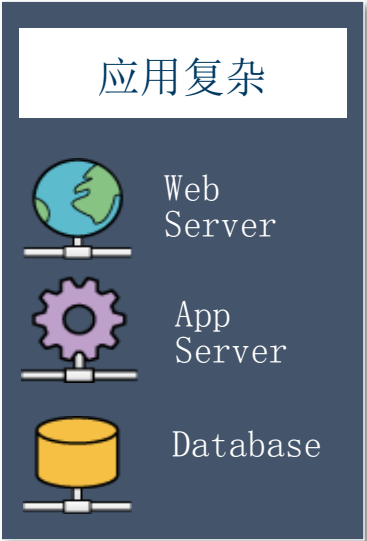
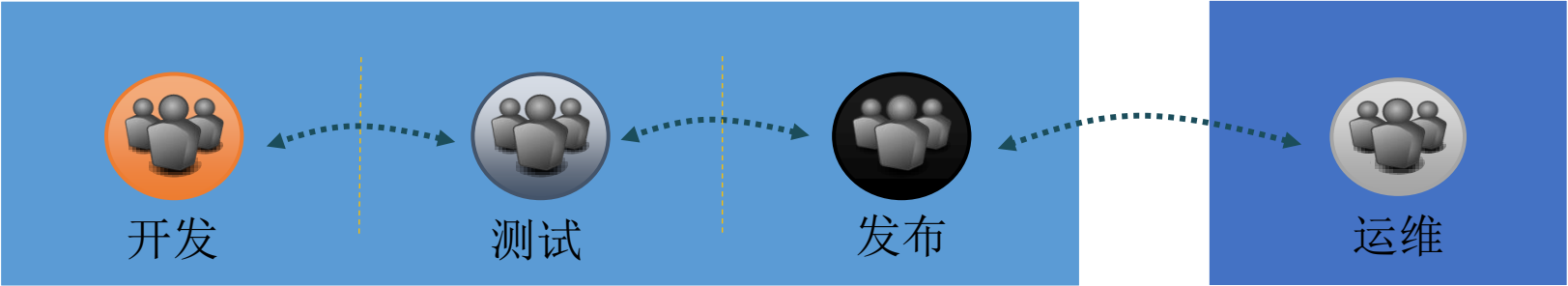
## 平衡速度、成本、质量和风险

软件交付过程自动化，以便消除人力/资源的浪费和工期延误。

## 降低客户反馈的时间

扩大客户反馈圈，以便可以持续提高

# 当前软件交付的复杂度



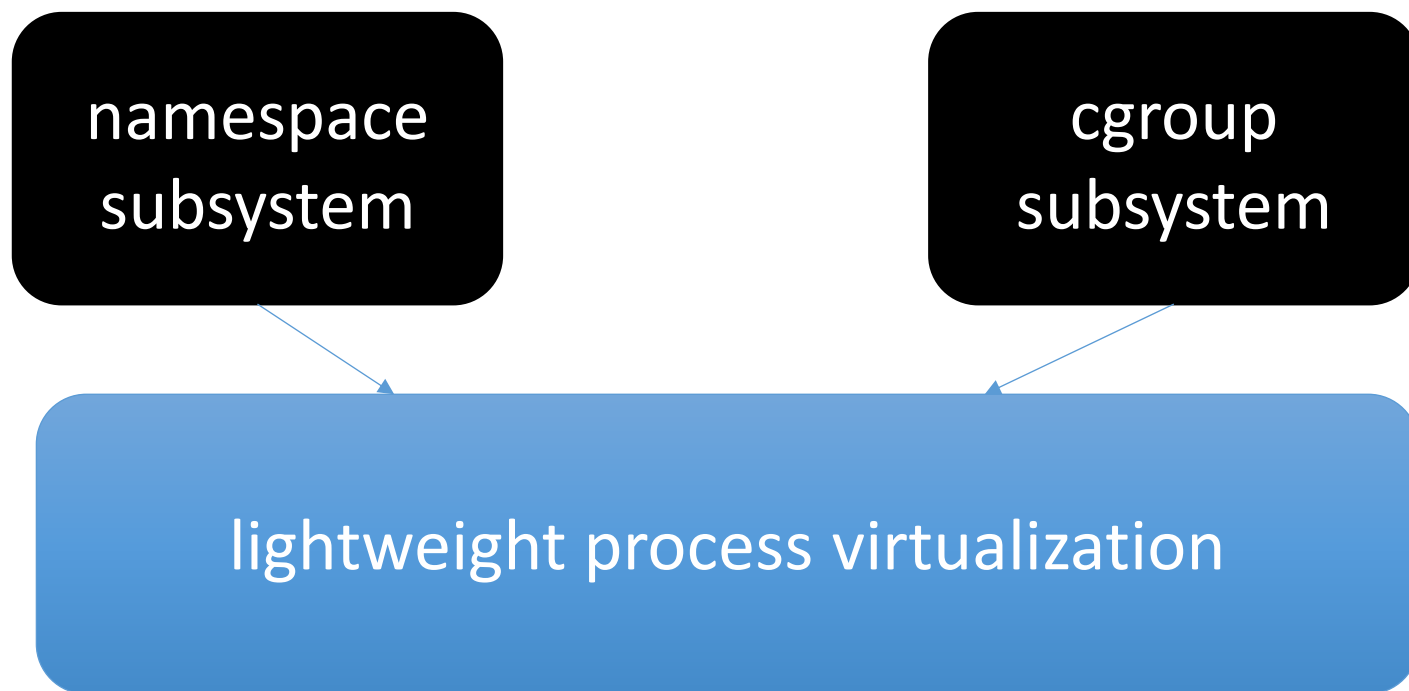


容器在开发中能做什么

# 容器的特性

- 容器技术--容器虚拟化技术--虚拟化技术
- Linux 内核原生提供特性

# 轻量级虚拟化技术



# Namespace 列表

Namespace	Constant	Isolates
Cgroup	<b>CLONE_NEWCGROUP</b>	Cgroup root directory
IPC	<b>CLONE_NEWIPC</b>	System V IPC, POSIX message queues
Network	<b>CLONE_NEWNET</b>	Network devices, stacks, ports, etc.
Mount	<b>CLONE_NEWNS</b>	Mount points
PID	<b>CLONE_NEWPID</b>	Process IDs
User	<b>CLONE_NEWUSER</b>	User and group IDs
UTS	<b>CLONE_NEWUTS</b>	Hostname and NIS domain name

# Namespace的系统调用函数

- **clone()** – 实现线程的系统调用，用来创建一个新的进程，并可以通过设计上述参数达到隔离。

```
int clone(int (*fn)(void *), void *child_stack, int flags, void *arg, ... /* pid_t  
*ptid, void *newtls, pid_t *ctid */ );
```

- **unshare()** – 使某进程脱离某个namespace

```
int unshare(int flags);
```

- **setns()** – 把某进程加入到某个namespace

```
int setns(int fd, int nstype);
```

# Clone UTS Namespace案例

```
[root@zabbix-server ~]# ls
anaconda-ks.cfg  nginx.conf  original-ks.cfg  test  test_namespace.c  test_namespace_uts  test_namespace_uts.c  test.
[root@zabbix-server ~]# ./test_namespace_uts
父进程 - 开启容器！
进入到分配的 Container!
[root@heidsoft ~]# ps -ef|grep test_namespace_uts
root      4580  4552  0 13:55 pts/0    00:00:00 ./test_namespace_uts
root      4597  4581  0 13:55 pts/0    00:00:00 grep --color=auto test_namespace_uts
[root@heidsoft ~]# ls
anaconda-ks.cfg  nginx.conf  original-ks.cfg  test  test_namespace.c  test_namespace_uts  test_namespace_uts.c  test.
[root@heidsoft ~]#
```

```
[root@heidsoft ~]# exit
exit
```

父进程 - 停止容器！

```
[root@zabbix-server ~]#
```

```
[root@zabbix-server ~]#
```

# Clone UTS Namespace 示例code

```
#include <sys/wait.h>
#include <stdio.h>
#include <sched.h>
#include <signal.h>
#include <unistd.h>

/* 定义一个给 clone 用的栈，栈大小 1M */
#define STACK_SIZE (1024 * 1024)
static char container_stack[STACK_SIZE];

char* const container_args[] = {
    "/bin/bash",
    NULL
};

int container_main(void* arg)
{
    printf("Container - inside the container!\n");
    /* 直接执行一个 shell，以便我们观察这个进程空间里的资源是否被隔离了 */
    execv(container_args[0], container_args);
    printf("Something's wrong!\n");
    return 1;
}

int main()
{
    printf("Parent - start a container!\n");
    /* 调用 clone 函数，其中传出一个函数，还有一个栈空间的（为什么传尾指针，因为栈是反着的） */
    int container_pid = clone(container_main, container_stack+STACK_SIZE, SIGCHLD, NULL);
    /* 等待子进程结束 */
    waitpid(container_pid, NULL, 0);
    printf("Parent - container stopped!\n");
    return 0;
}
```

# cgroup subsystem 来源

- cgroups（Control Groups）最初叫Process Container，由Google工程师（Paul Menage和Rohit Seth）于2006年提出，后来因为Container有多重含义容易引起误解，就在2007年更名为Control Groups，并被整合进Linux内核。



# cgroup subsystem 作用

特性	描述
资源限制（Resource Limitation）	<b>cgroups</b> 可以对进程组使用的资源总额进行限制。如设定应用运行时使用内存的上限，一旦超过这个配额就发出 <b>OOM（Out of Memory）</b> 。
优先级分配（Prioritization）	通过分配的 <b>CPU</b> 时间片数量及硬盘 <b>IO</b> 带宽大小，实际上就相当于控制了进程运行的优先级。
资源统计（Accounting）	<b>cgroups</b> 可以统计系统的资源使用量，如 <b>CPU</b> 使用时长、内存用量等等，这个功能非常适用于计费。
进程控制（Control）	<b>cgroups</b> 可以对进程组执行挂起、恢复等操作。

# Linux 内核中cgroup资源控制源码文件

Name	Kernel Object name	Module
blkio	<b><i>io_cgrp_subsys</i></b>	block/blk-cgroup.c
cpuacct	<b><i>cpuacct_cgrp_subsys</i></b>	kernel/sched/cpuacct.c
cpu	<b><i>cpu_cgrp_subsys</i></b>	kernel/sched/core.c
cpuset	<b><i>cpuset_cgrp_subsys</i></b>	kernel/cpuset.c
devices	<b><i>devices_cgrp_subsys</i></b>	security/device_cgroup.c
freezer	<b><i>freezer_cgrp_subsys</i></b>	kernel/cgroup_freezer.c
hugetlb	<b><i>hugetlb_cgrp_subsys</i></b>	mm/hugetlb_cgroup.c
memory	<b><i>memory_cgrp_subsys</i></b>	mm/memcontrol.c
net_cls	<b><i>net_cls_cgrp_subsys</i></b>	net/core/netclassid_cgroup.c
net_prio	<b><i>net_prio_cgrp_subsys</i></b>	net/core/netprio_cgroup.c
perf_event	<b><i>perf_event_cgrp_subsys</i></b>	kernel/events/core.c
pids	<b><i>pids_cgrp_subsys</i></b>	kernel/cgroup_pids.c

# 容器在开发中能做什么

- 替代vagrant,用它当做轻量级虚拟机使用
- 替代vcenter 模板,用容器镜像,当做模板使用
- 替代rpm安装mysql应用,用容器安装mysql 环境
- 替代jdk安装,用容器直接部署jdk 环境
- 替代python virtualenv,用容器直接搭建不同版本的python开发环境
- and so on ...

我在开发工作中如何使用容器

# 分阶段使用，逐步深入

## 第三阶段

服务型使用，将工具以服务形式呈现

- 1.将工具能力平台化，用平台呈现工具服务
- 2.将工具结合更多服务场景，提升工具服务能力

- 使用蓝鲸平台能力
- 构建服务化
- saas应用

## 第二阶段

管理型使用，将工具与流程结合

- 1.用工具将流程打通，是流程是动起来
- 2.用工具将流程中使用的资源管理好

- Docker
- Git
- Jenkins
- Cmdb
- Zabbix

## 第一阶段

学习型使用，了解基本概念，掌握基本原理

- 1.以独立工具形式使用
- 2.知道怎么用，怎么玩

- Docker
- Code Project

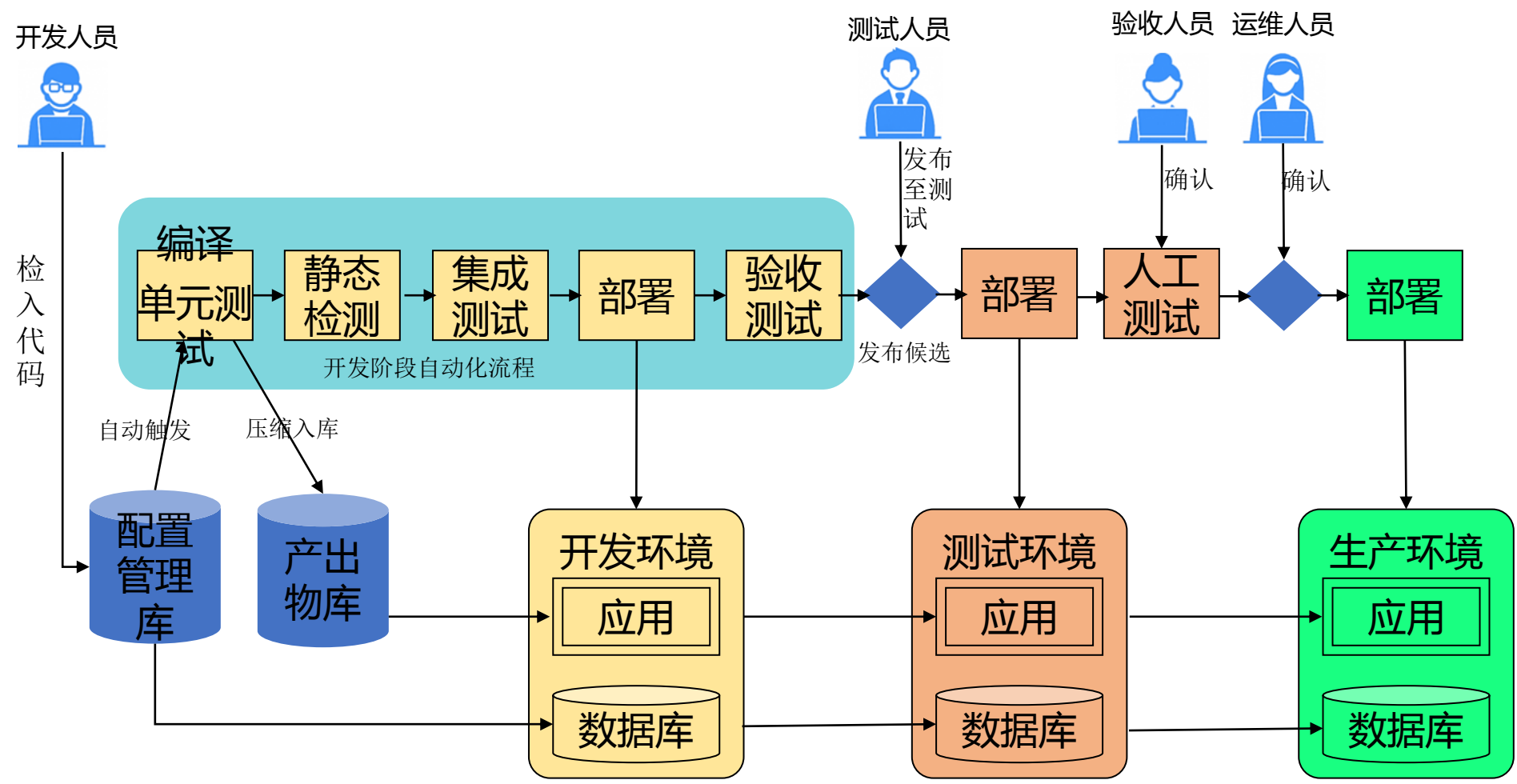
# 第一阶段:用容器构建环境

- 用容器统一我的开发环境
- 用容器统一我的测试环境
- 用容器统一我的发布环境
- 用容器统一我的生产环境

## 第二阶段:管理好容器资源，让容器服务开发流程

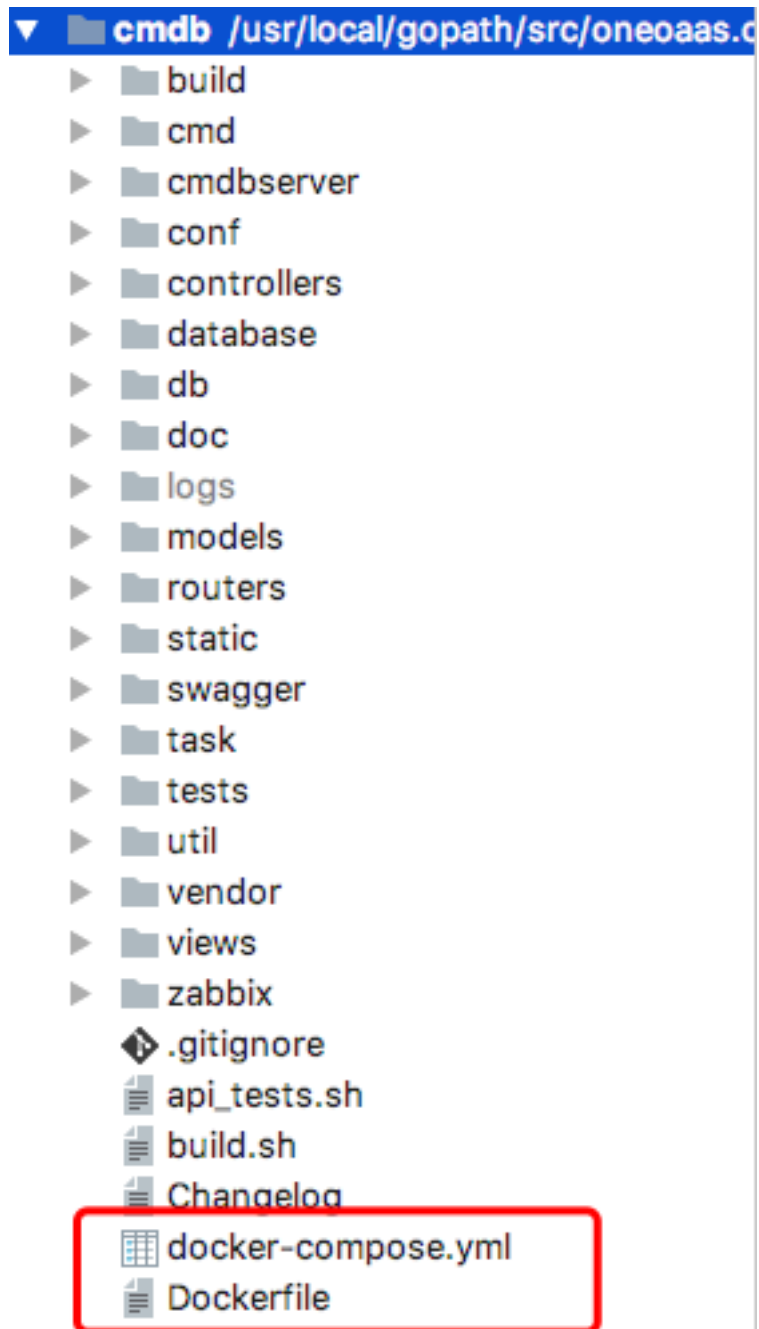
- 持续集成中，主动应用容器
- 持续交付中，管理好容器

# 集成到交付的一般流程

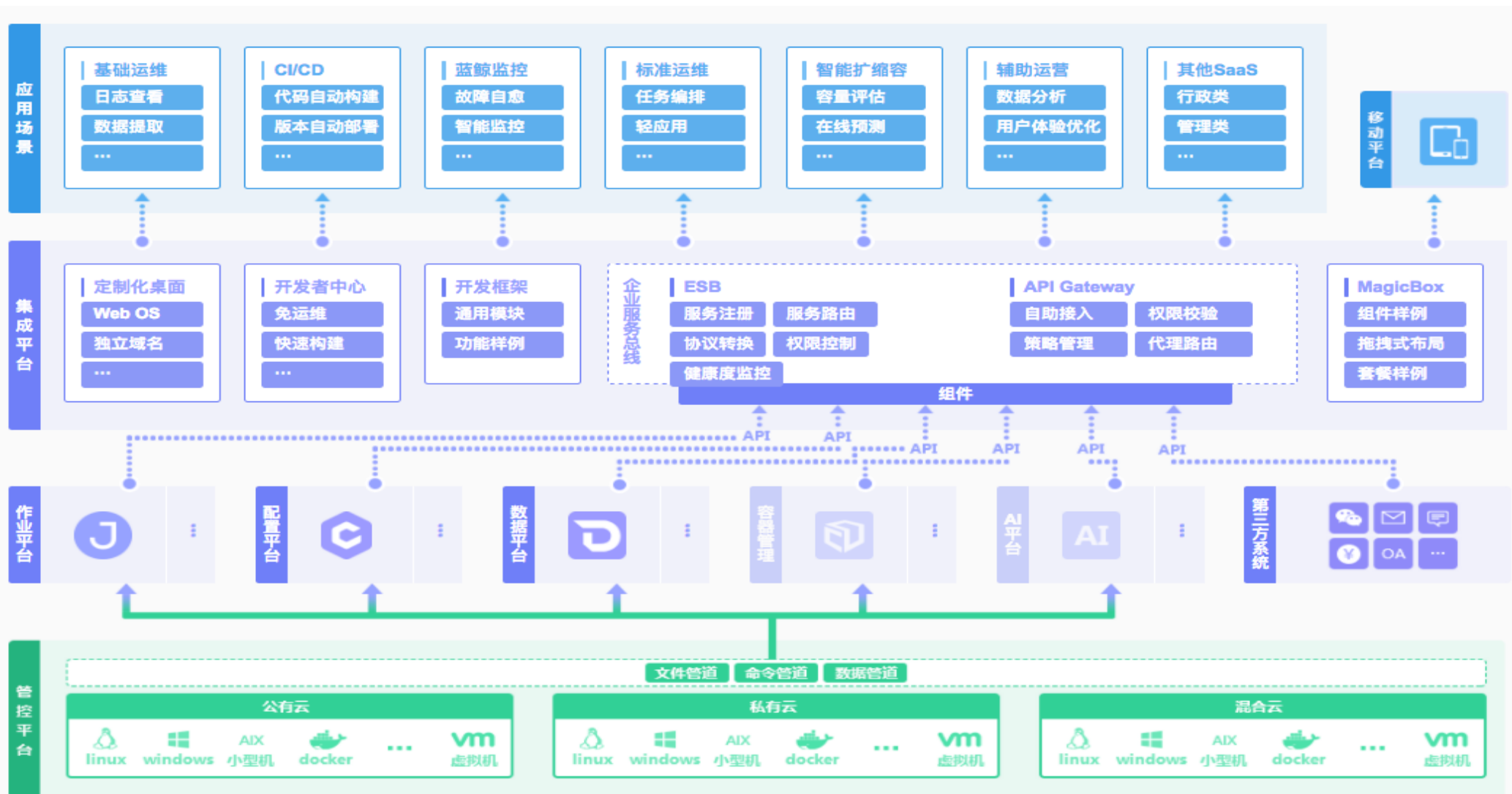




# 工程案例




# 第三阶段:用容器服务化应用工具



# 蓝鲸官方saas应用

官方SaaS



配置平台   
蓝鲸智云

立即体验



作业平台   
蓝鲸智云

立即体验



Agent安装   
蓝鲸智云


立即体验



标准运维   
蓝鲸智云

立即体验



开发者中心   
蓝鲸智云


立即体验



鹰眼   
蓝鲸智云

立即体验



开发框架   
蓝鲸智云

立即体验



作业平台移动版   
蓝鲸智云

立即体验

# 基于蓝鲸平台开发监控saas应用



## 关于OneOaaS

OneOaaS为用户提供**运维工具，解决方案。包括CMDB，监控系统，代码部署等云时代的运维解决方案。**

公司由一批资深运维专家组成,他们对云计算和自动化运维有着独特的见解, 对技术有着狂热的追求，对各行业业务有着透彻的理解，能够为用户提供切实有效的解决方案，并善于为客户解决运维难题。

其提倡用理论指导运维的方式，帮助客户建立运维意识，制定运维规范，使用成熟高效的运维工具去迎接大规模运维问题。公司自研的运维工具，能够有效解决运维中的**资产管理问题，配置管理问题，开发测试难题，代码管理问题，监报告警问题。****真正的做到运筹帷幄之中，决胜千里之外。**

公司官方网站: <http://www.oneoaas.com>



谢谢聆听