Gdevops

全球敏捷运维峰会

持续交付：新产品的成长思维

演讲人：黎嘉豪

自我介绍

- 百度工程效率部 – 技术教练
- 工程师，负责研发工具链、持续交付、DevOps
- 百度学院高级讲师
- 『百度方法+』持续交付专题作者

- 云计算研发高级工程师

互联网时代对软件交付的诉求

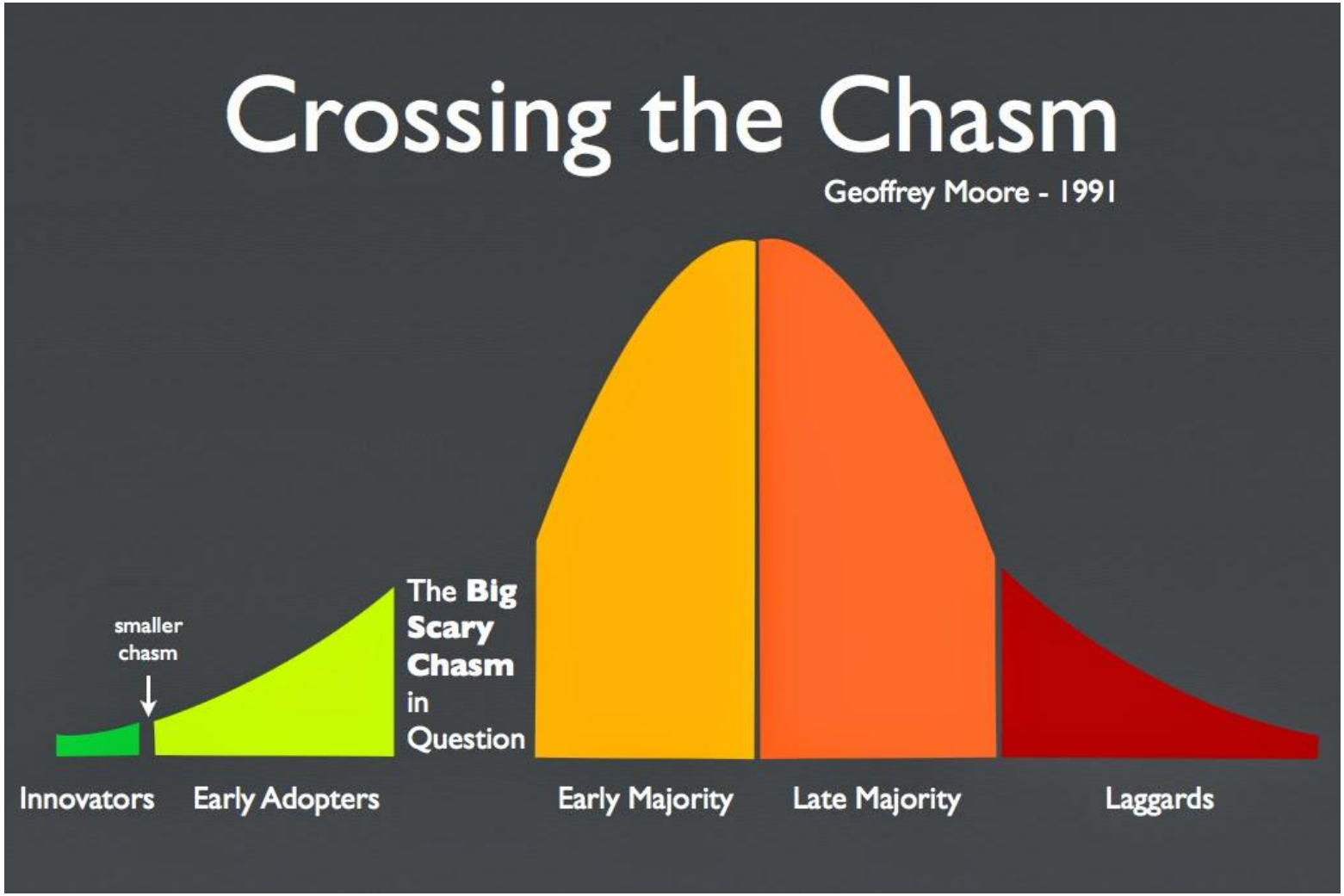
我们处在一个VUCA的时代

软件交付面临易变性、不确定性、复杂性、模糊性

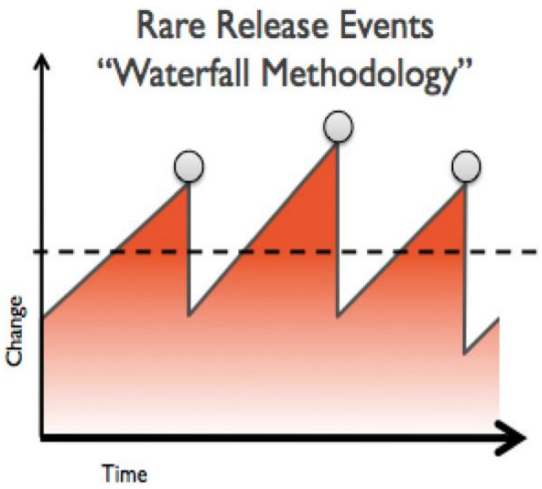
- ✓ 专注、极致、口碑、快
- ✓ 快速迭代
- ✓ 快速转型
- ✓ 只有第一，没有第二

- ✓ 用户体验至上
- ✓ 细节决定成败
- ✓ 缺陷 -> 差评 -> 卸载
- ✓ 服务故障 ->市场丢失

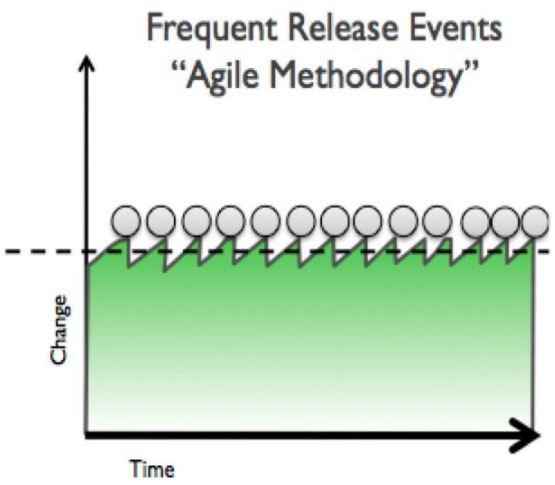
团队的挑战


Crossing the Chasm
Geoffrey Moore - 1991
smaller chasm
The Big Scary Chasm in Question
Innovators
Early Adopters
Early Majority
Late Majority
Laggards

团队的挑战



Effort Peaks
High Risk



Smoother Effort
Less Risk

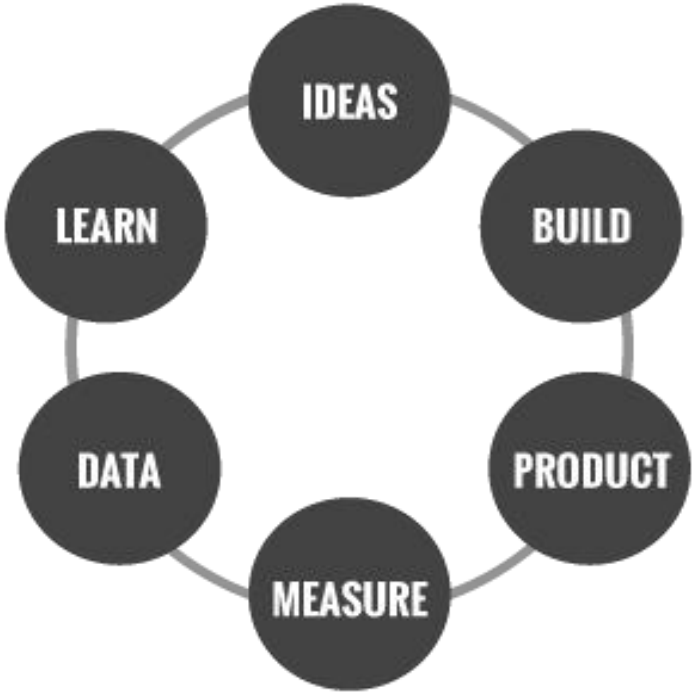
快速上线

从需求到交付端到端交付周期缩短

质量保证

效率提升的同时，保证质量稳定

团队的挑战



精益创业的产品开发方式

需要

不断 + 低成本 + 快速 验

证想法

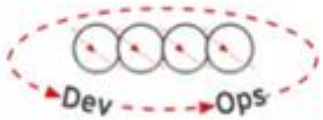
团队的挑战

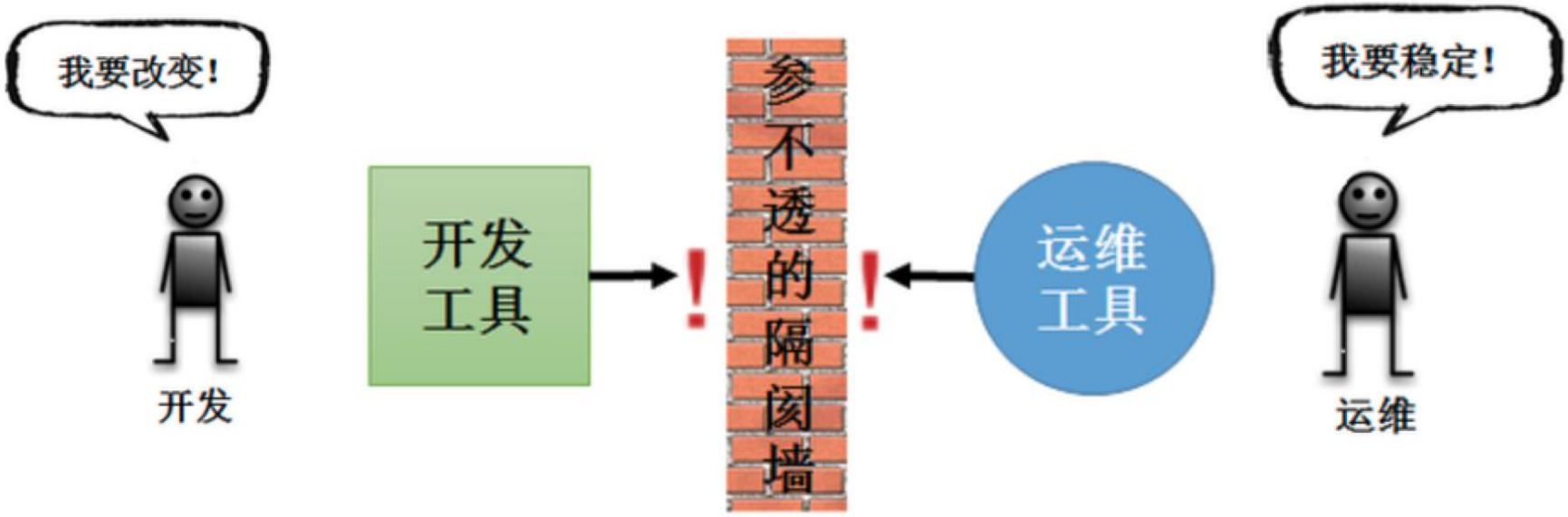
人员、流程、技术被「墙」阻断，Throw it over the wall...

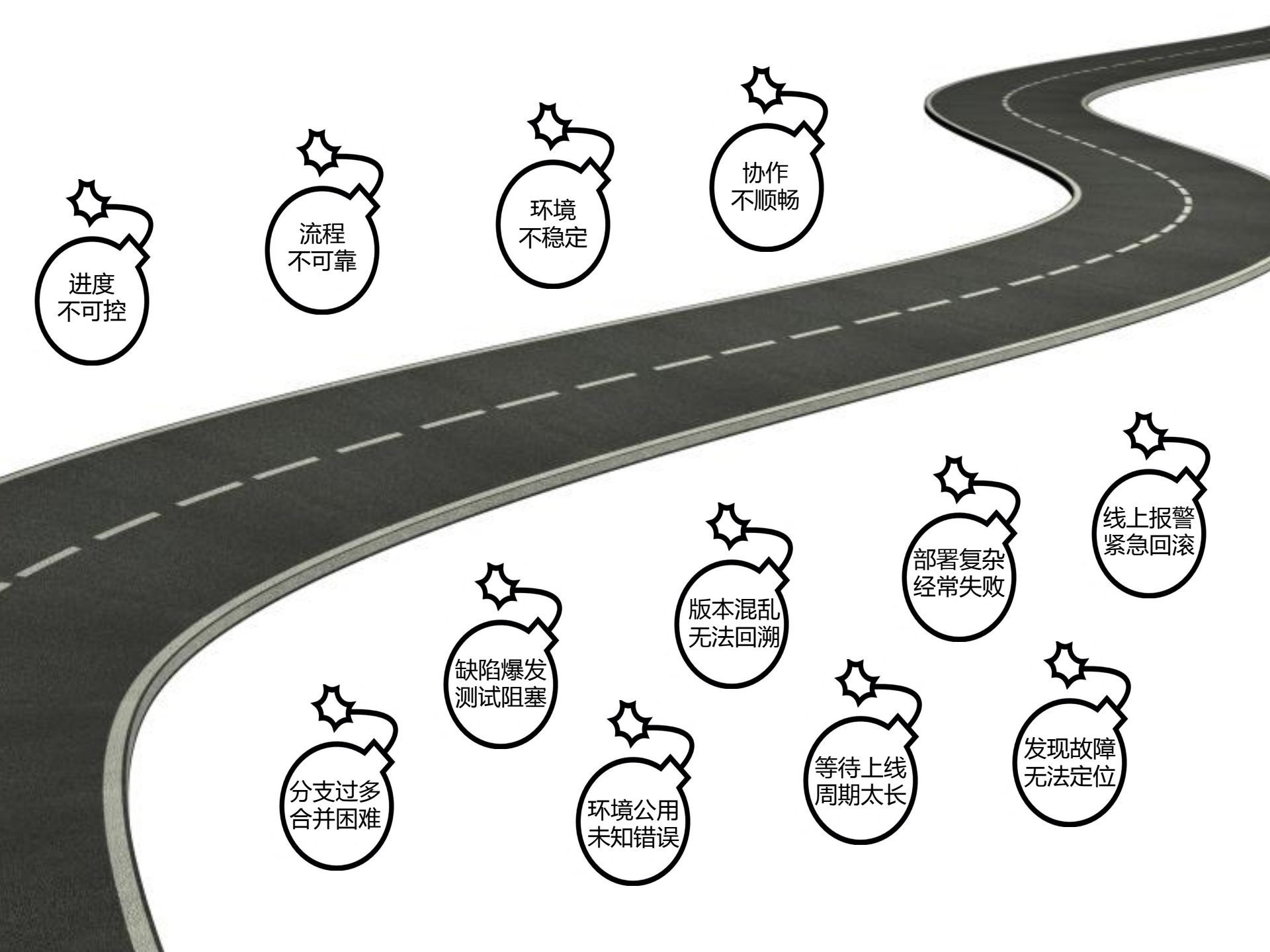
开发领域

运维领域

Development Process	Application Architecture	Deployment & Packaging	Application Infrastructure
Waterfall	Monolithic	Physical Servers	Datacenter
Agile	N-Tier	Virtual Servers	Hosted
DevOps	**Microservices**	**Containers**	**Cloud**




我要改变!
开发
开发工具
!
参不透的隔阂墙
!
运维工具
我要稳定!
运维


进度
不可控
流程
不可靠
环境
不稳定
协作
不顺畅
线上报警
紧急回滚
部署复杂
经常失败
版本混乱
无法回溯
缺陷爆发
测试阻塞
分支过多
合并困难
环境公用
未知错误
等待上线
周期太长
发现故障
无法定位

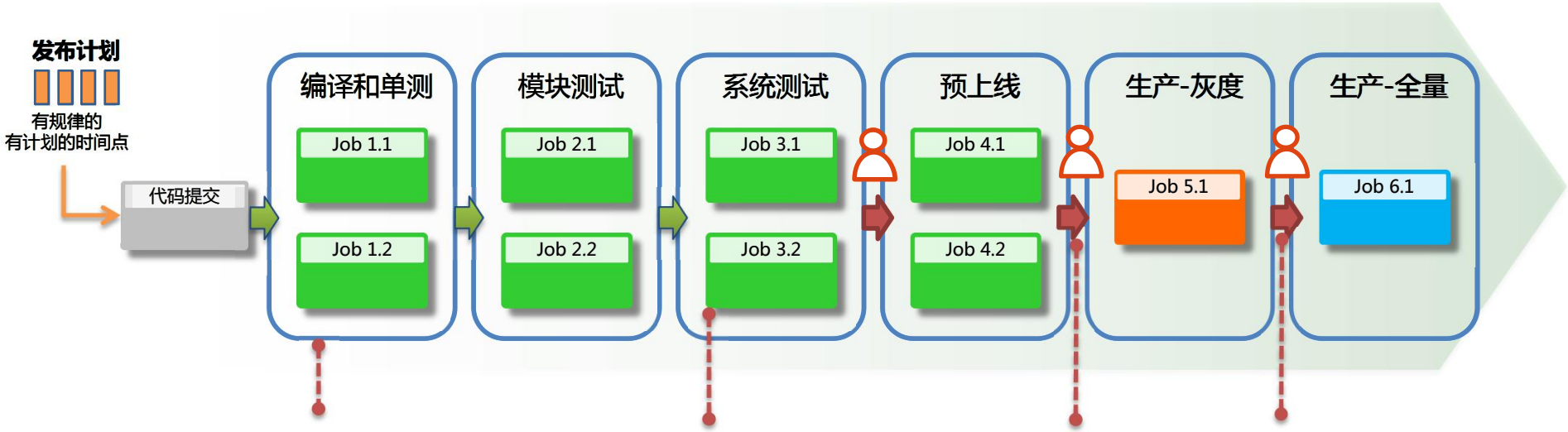
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software

1st principle of the Agile Manifesto

我们的目标

1. 缩短提交代码 → 部署上线时间
2. 快速得到反馈
3. 软件处于随时都可以部署的状态
4. 可将任意发布的版本、部署在任意环境中
5. 交付过程：可靠、可预期、可视化

可靠可重复的流水线



划分阶段（Stage）：

- 多个Stages是串行执行
- 前一个Stage成功完成后自动触发
- 也可以通过手工触发

执行作业（Job ）：

- Stage里的多个Jobs串/并行执行
- Stage里的多个Jobs定时执行
- 自动判断或人工标记Pass/Fail
- Job Fails → Stage Fails

质量门（通过标准）：

- Pass/Fail判定标准
- 测试通过率>xx%
- 代码覆盖率>xx%

决策点（人工干预）：

- 可配置人工决策，一键Approve后流水线自动执行
- 也可配置前序Stage成功后自动触发执行

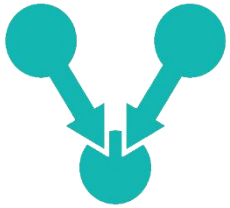
组成流水线的元素

- 记录项目中的一切变化
- 加速构建速度，管理模块依赖
- 让软件随时可以运行
- 分级测试，建立快速反馈环
- 弹性灵活的基础设施
- 无差异的部署过程，减少上线风险

全面的配置管理

版本控制

源代码、测试代码
数据库脚本
构建、部署脚本
文档、库文件



依赖管理

组件管理
外部库文件管理

软件配置管理

构建、部署、运行
所需的配置文件

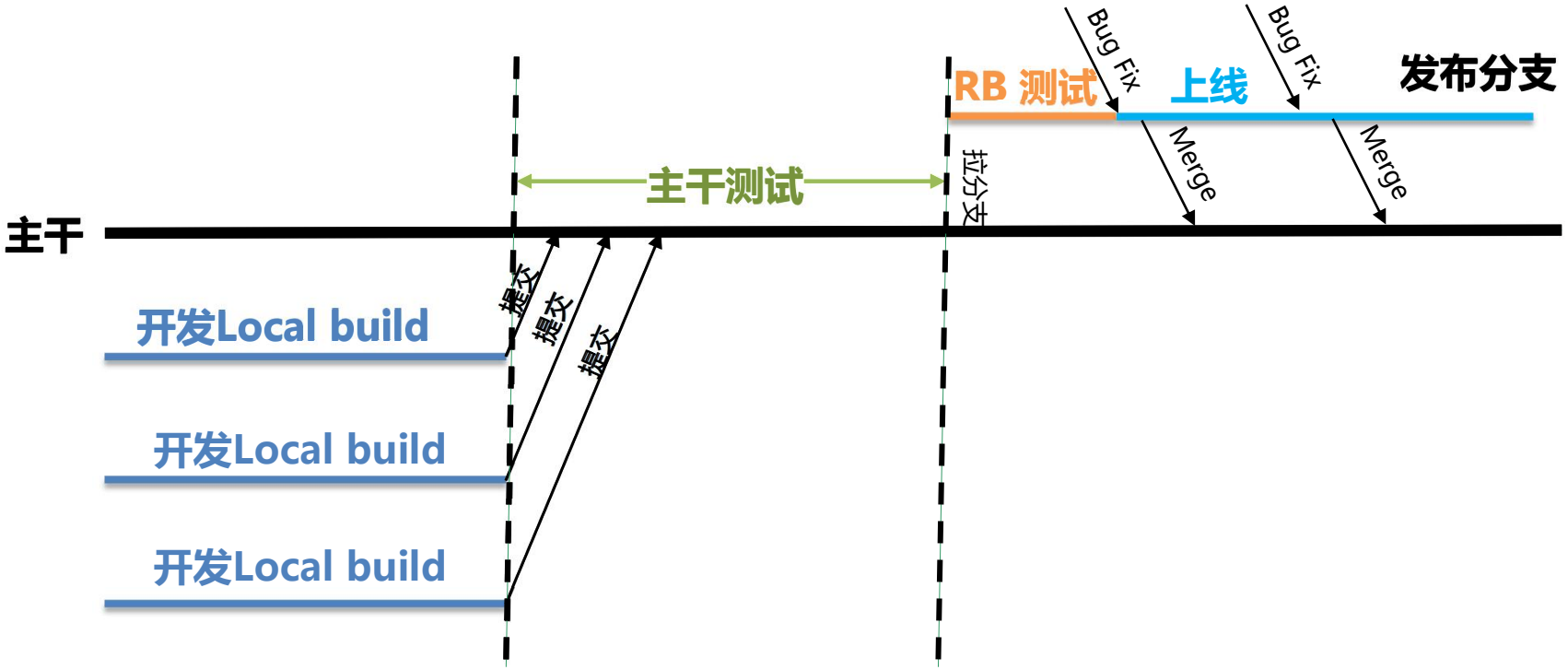
环境管理

基础设施
操作系统
运行软件依赖工具、软件

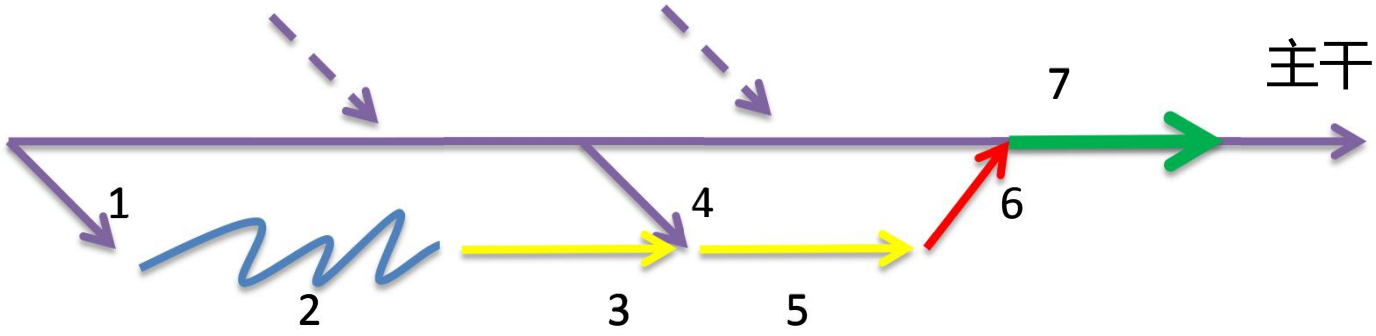
配置管理 - 分支策略

主干开发、分支发布

- 主干开发，自动化验证通过后，拉出Release Branch分支提测
- QA在RB上执行补充测试、如有bug则修复，并且同步到主干，基于分支发布版本
- 上线后发现Bug， 基于从上次上线的分支进行开发和发布 ，待上完线后将代码Merge回主干

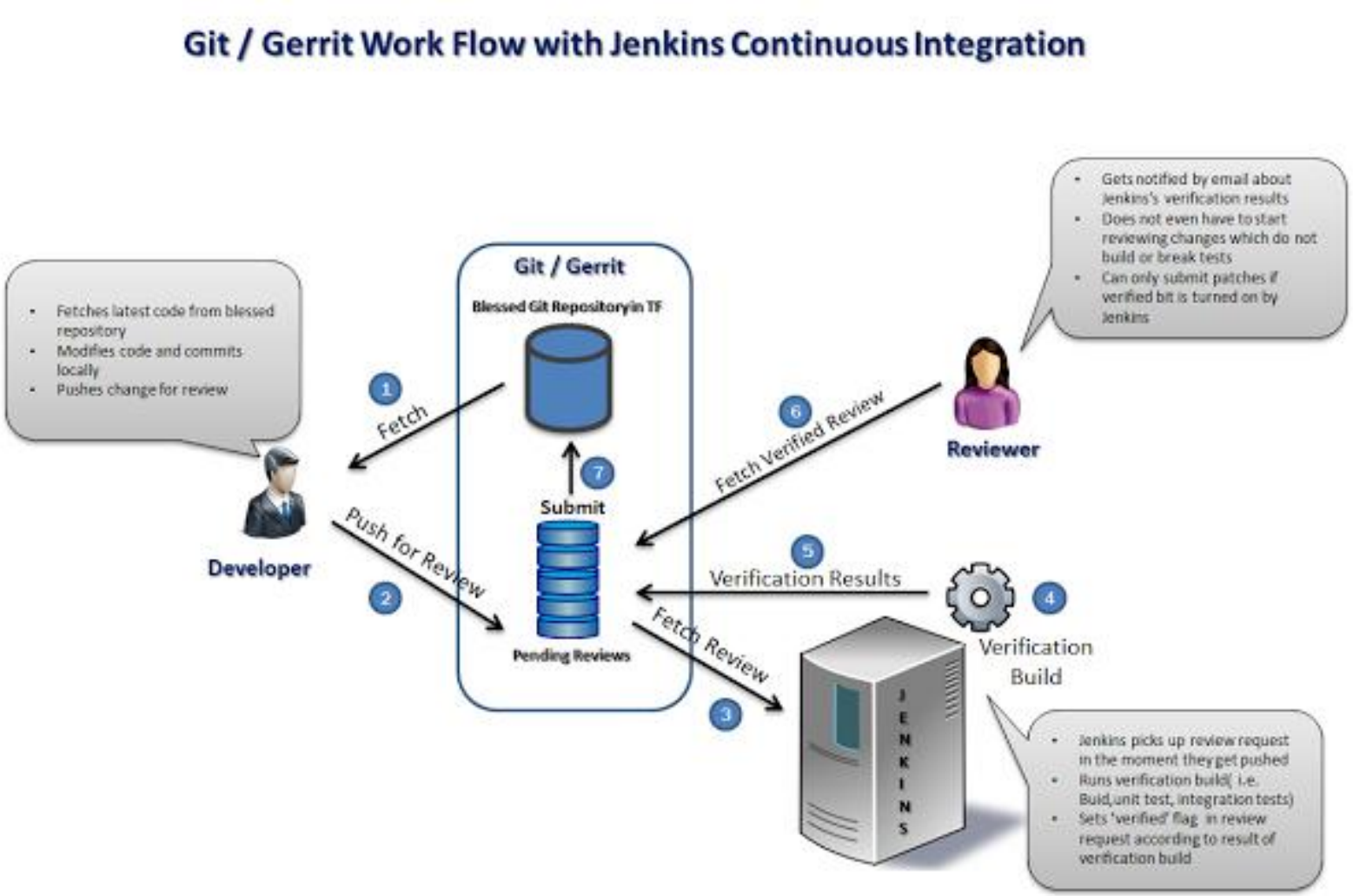


配置管理 – Check In Dance

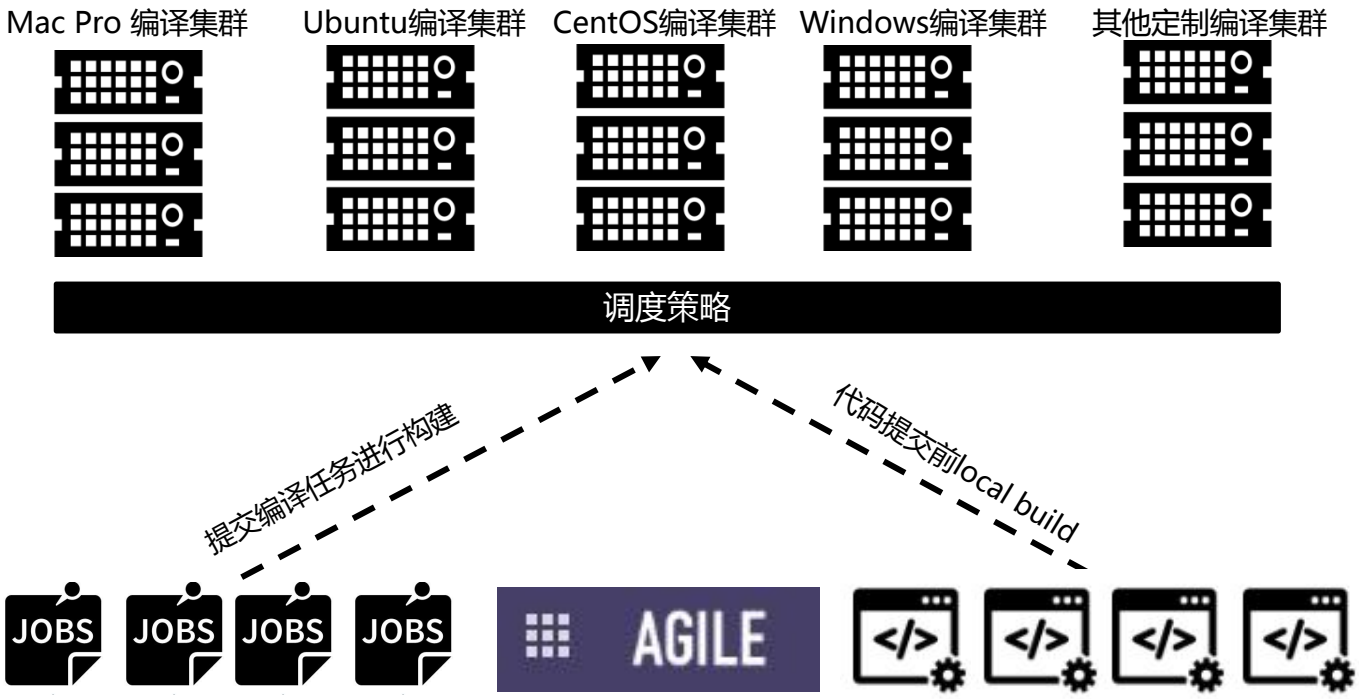


1. 从稳定的最新版本中 Check out 代码
2. 在本地开发新的功能
3. 执行本地构建（测试，代码检查）
4. 再次Check out 代码
5. 执行本地构建（测试，代码检查）
6. 提交代码至当前主干
7. 执行主干上的编译，等待持续集成结果
8. 如果通过则结束，没有通过则跳转到步骤（2）

配置管理 – Gerrit Workflow

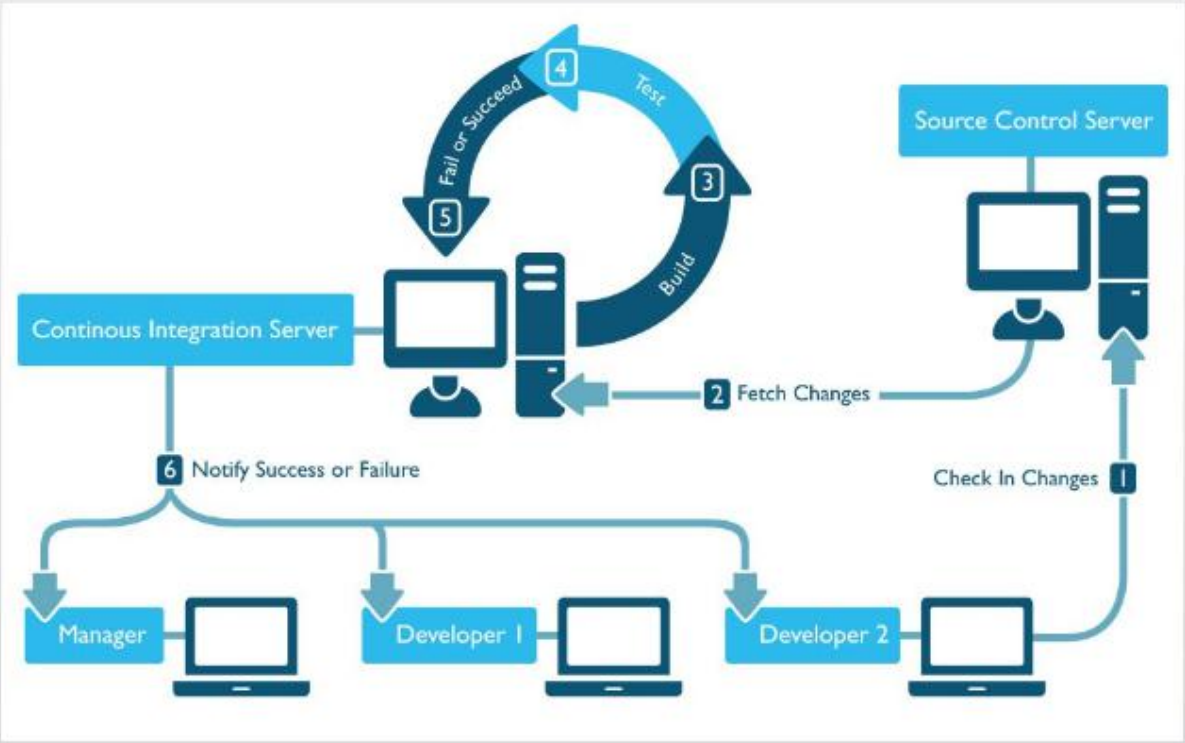


构建管理



- 集群加速，提高并发场景吞吐量
- 机器针对编译任务硬件优化
- 构建任务从Push 改成 Pull
- 模块间依赖采用二进制构建产物
- 1-5-10 法则
- 编译结果推送至『产品仓库』管理

持续集成



坚持的原则：

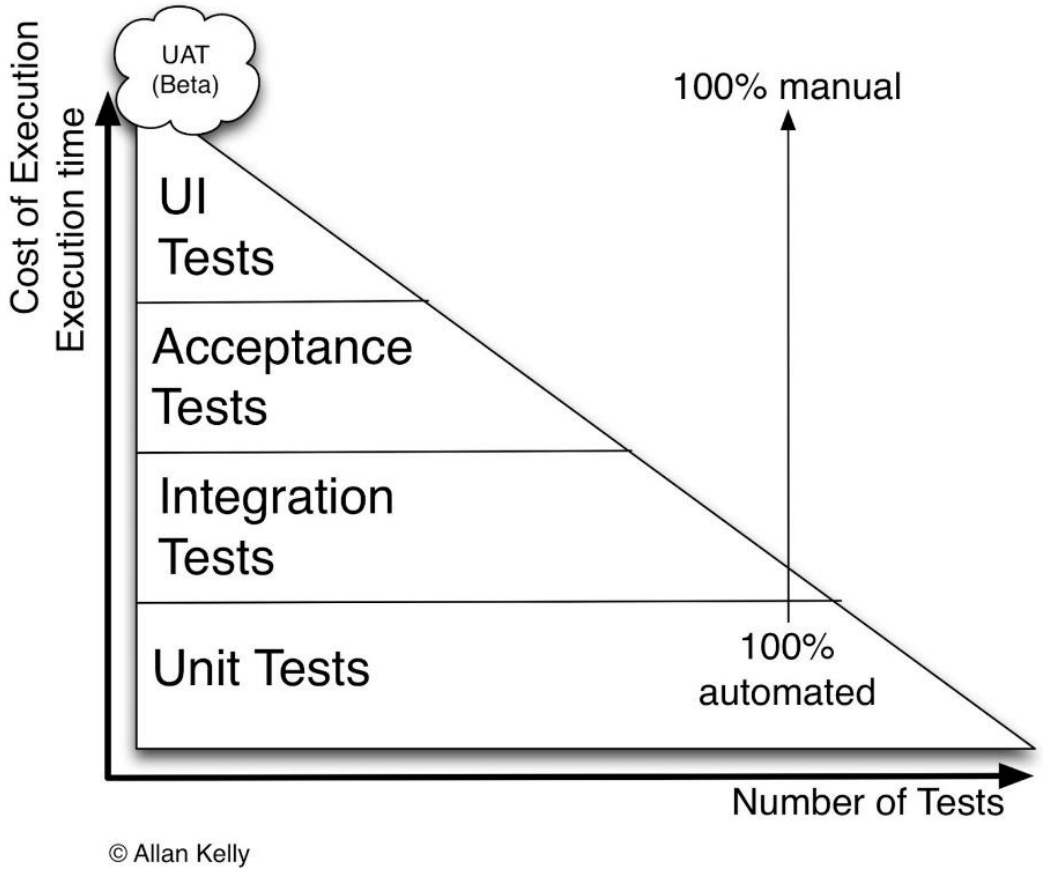
- 频繁提交
- 主干上做持续集成
- 至少每天进行集成
- 自动化构建和测试
- 分级测试快速反馈
- 红灯需要立即修复

测试管理

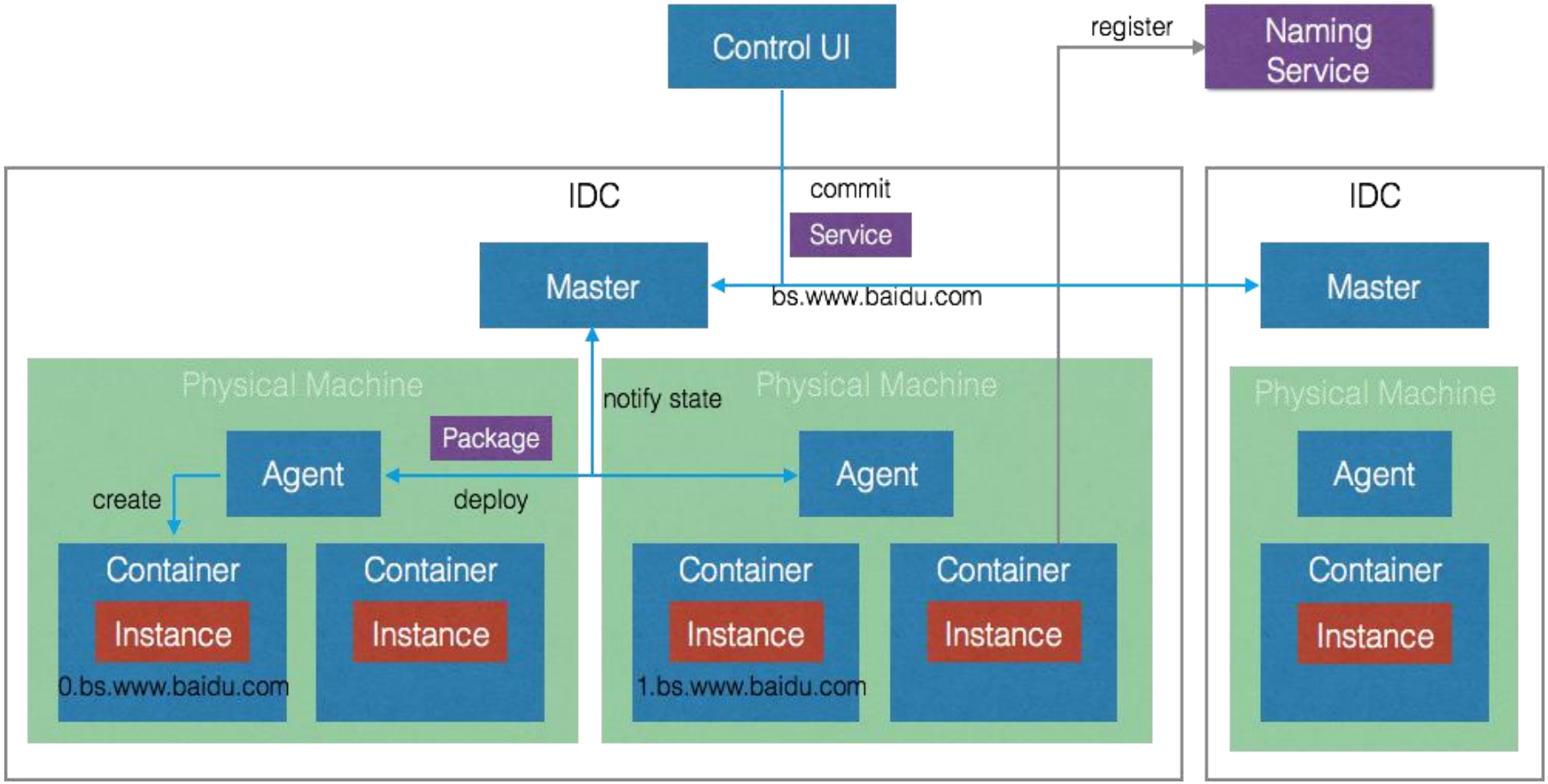
建立分级测试体系，从多个层次和多个验证角度实现质量防护网

测试管理

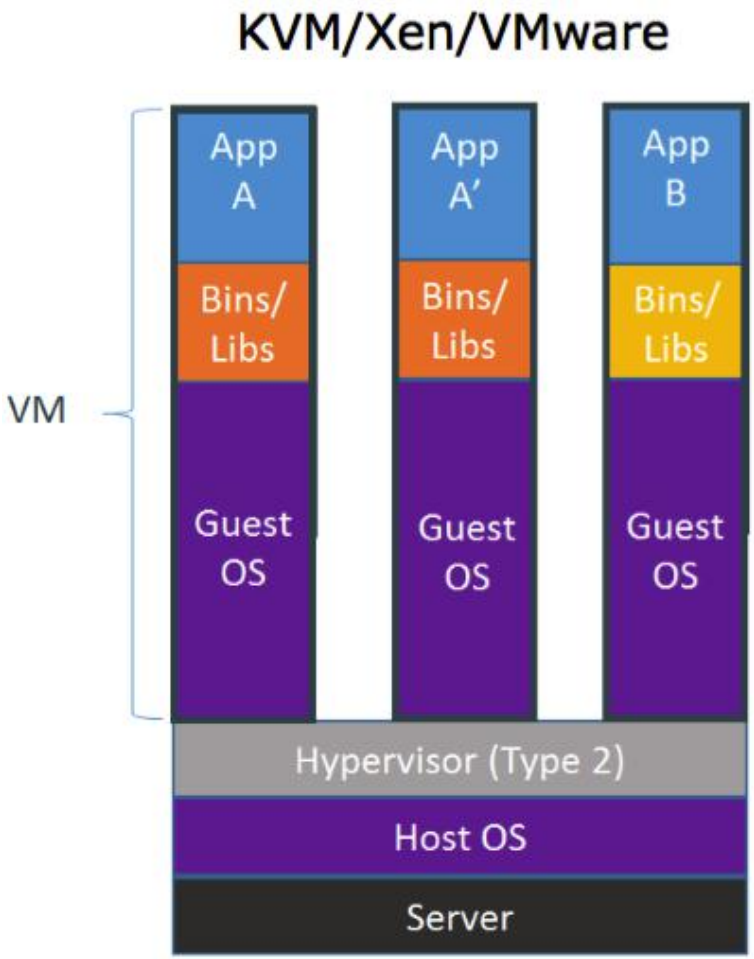
建立分级测试体系，保持编译和测试结果能快速反馈

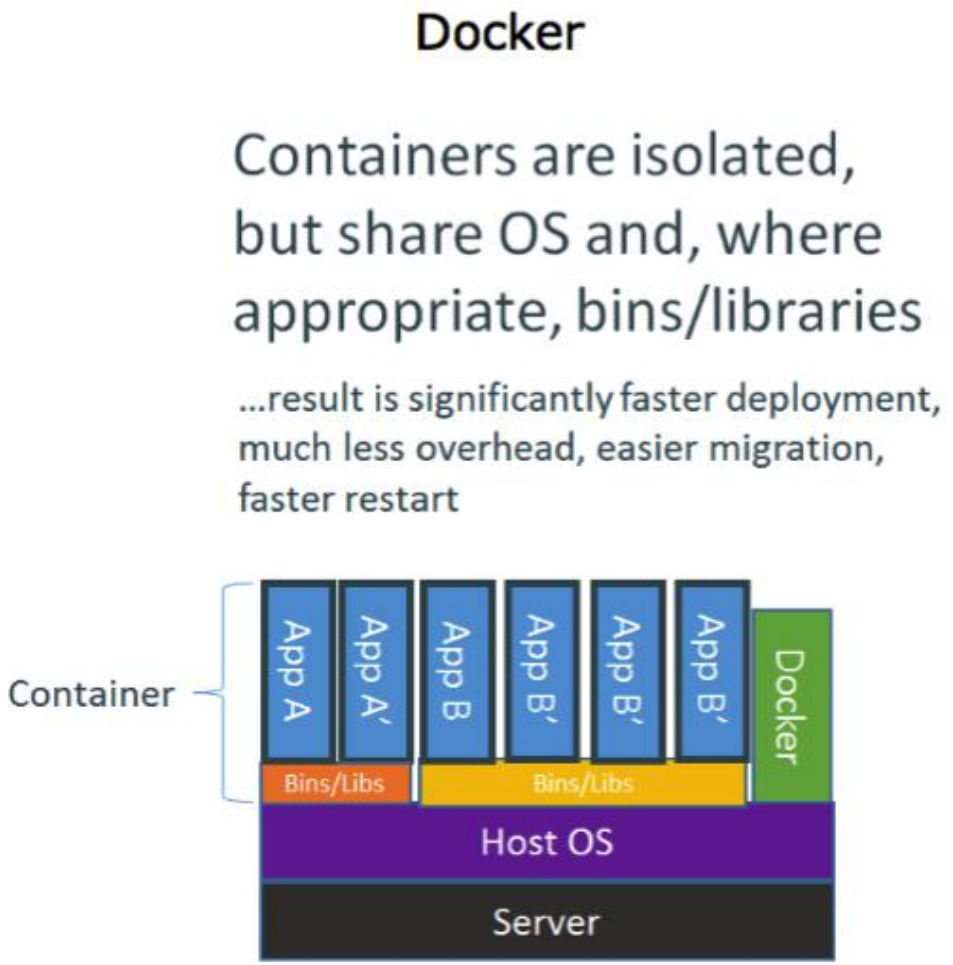


环境管理：容器技术


Control UI
register
Naming Service
IDC
commit
Service
Master
bs.www.baidu.com
IDC
Master
Physical Machine
notify state
Physical Machine
Physical Machine
Package
Agent
Agent
Agent
create
deploy
Container
Instance
0.bs.www.baidu.com
Container
Instance
Container
Instance
1.bs.www.baidu.com
Container
Instance
Container
Instance

环境管理：容器与虚拟化


KVM/Xen/VMware
VM
App A
Bins/ Libs
Guest OS
App A'
Bins/ Libs
Guest OS
App B
Bins/ Libs
Guest OS
Hypervisor (Type 2)
Host OS
Server


Docker
Containers are isolated, but share OS and, where appropriate, bins/libraries
...result is significantly faster deployment, much less overhead, easier migration, faster restart
Container
App A
App A'
App B
App B'
App B'
App B'
Docker
Bins/Libs
Bins/Libs
Host OS
Server

部署管理 - 服务描述

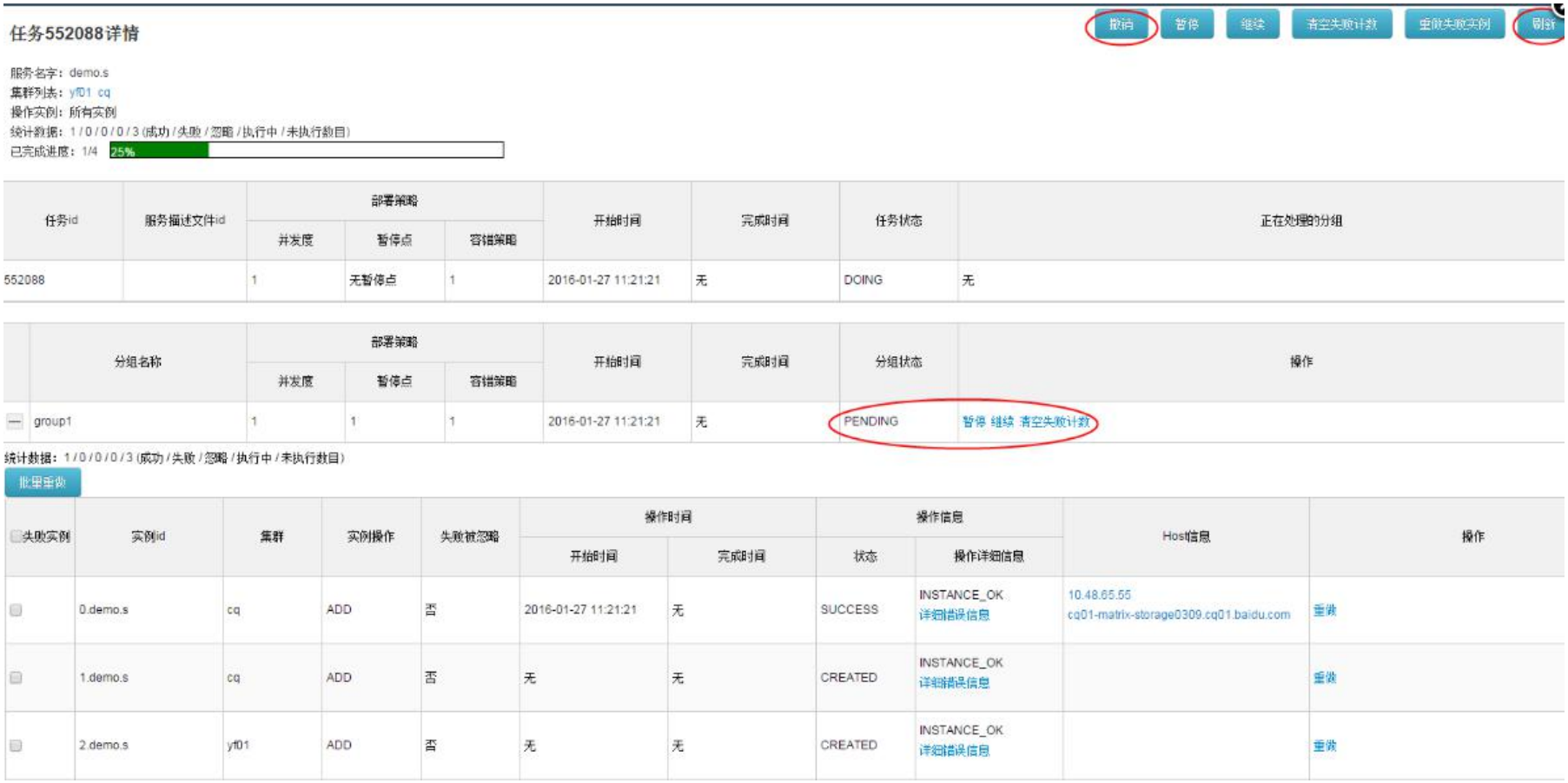
服务描述包含两部分：

- **Service描述**
  - Package
  - Server个数
  - 集群分布
- **Operation描述**
  - 上线过程的并发控
  - 部署暂停点
  - 容错处理

```
package:
    type:       LITE
    url:        ftp://cp01-cos-dev01
    version:    1.0
resource:
    cpu:
        numCores:           10
    memory:
        sizeMB:             1024
    network:
        inBandwidthMBPS:    10
        outBandwidthMBPS:   10
    process:
        numThreads:         256
    port:
        rangeLimit:
            min:            8000
            max:            9000
        dynamic:            [http_po
    workspace:
        sizeMB:             10240
        inode:              10000
```

```
operation:
    operationTree:
        name: root
        policy:
            concurrency:
                count: 2
            planPause:
                counts:
                    - 1
            failPause:
                count: 1
        serverOffsets: 0-3
```

部署管理 - 灰度发布



任务552088详情

撤消 暂停 继续 清空失败计数 重做失败实例 刷新

服务名字：demo.s
集群列表：yf01 cq
操作实例：所有实例
统计数据：1 / 0 / 0 / 0 / 3 (成功 / 失败 / 忽略 / 执行中 / 未执行数目)
已完成进度：1/4 25%

任务id	服务描述文件id	部署策略			开始时间	完成时间	任务状态	正在处理的分组
		并发度	暂停点	容错策略				
552088		1	无暂停点	1	2016-01-27 11:21:21	无	DOING	无

	分组名称	部署策略			开始时间	完成时间	分组状态	操作
		并发度	暂停点	容错策略				
—	group1	1	1	1	2016-01-27 11:21:21	无	PENDING	暂停 继续 清空失败计数

统计数据：1 / 0 / 0 / 0 / 3 (成功 / 失败 / 忽略 / 执行中 / 未执行数目)

批量重做

失败实例	实例id	集群	实例操作	失败被忽略	操作时间		操作信息		Host信息	操作
					开始时间	完成时间	状态	操作详细信息		
	0.demo.s	cq	ADD	否	2016-01-27 11:21:21	无	SUCCESS	INSTANCE_OK 详细错误信息	10.48.65.55 cq01-matrix-storage0309.cq01.baidu.com	重做
	1.demo.s	cq	ADD	否	无	无	CREATED	INSTANCE_OK 详细错误信息		重做
	2.demo.s	yf01	ADD	否	无	无	CREATED	INSTANCE_OK 详细错误信息		重做

最佳实践汇总

配置管理

- 主干开发，分支发布
- 统一代码树结构
- 代码提交Check In Dance
- 代码、配置、数据分离
- 部署包标准化

构建管理

- 只生成一次二进制文件
- 任何时候可以重新构建二进制文件
- 制品仓库和数据管理
- 统一的依赖管理
- 标准化构建环境和平台

持续集成

- 每人每天提交
- 每天多次集成
- 质量内建原则
- 自动化构建和测试
- Test-Driven Development

测试管理

- 分级测试模型
- 原子性测试，顺序无关
- 维护最小测试数据集
- 流水线质量门和晋级标准

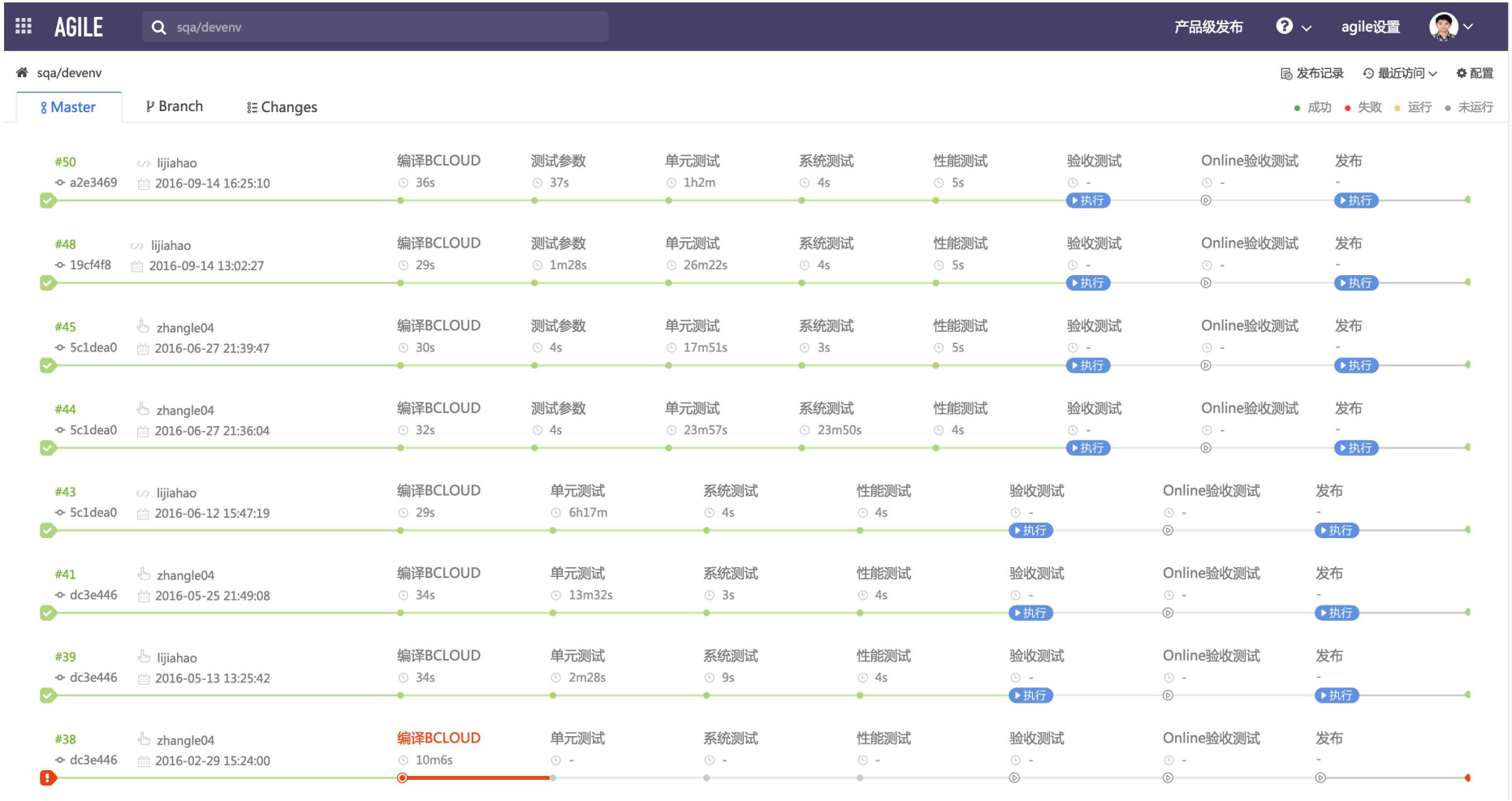
环境管理

- 标准化生产环境
- 基础设置即代码
- 禁止手工非受控方式进行变更环境
- 一键式重建测试、生产环境
- 自动故障迁移，自动扩容/缩容

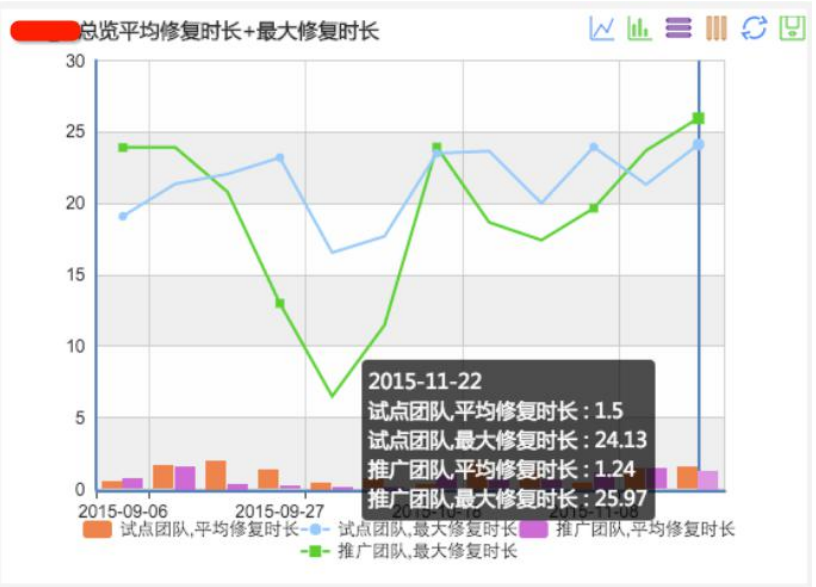
部署管理

- 统一部署平台
- 一键式部署、回滚
- 热部署，灰度发布
- 运维监控和业务监控

交付流水线


AGILE
sqa/devenv
产品级发布
agile设置
sqa/devenv
发布记录
最近访问
配置
Master
Branch
Changes
成功
失败
运行
未运行
#50 lijiahao a2e3469 2016-09-14 16:25:10 编译BCLOUD 36s 测试参数 37s 单元测试 1h2m 系统测试 4s 性能测试 5s 验收测试 - 执行 Online验收测试 - 发布 - 执行
#48 lijiahao 19cf4f8 2016-09-14 13:02:27 编译BCLOUD 29s 测试参数 1m28s 单元测试 26m22s 系统测试 4s 性能测试 5s 验收测试 - 执行 Online验收测试 - 发布 - 执行
#45 zhangle04 5c1dea0 2016-06-27 21:39:47 编译BCLOUD 30s 测试参数 4s 单元测试 17m51s 系统测试 3s 性能测试 5s 验收测试 - 执行 Online验收测试 - 发布 - 执行
#44 zhangle04 5c1dea0 2016-06-27 21:36:04 编译BCLOUD 32s 测试参数 4s 单元测试 23m57s 系统测试 23m50s 性能测试 4s 验收测试 - 执行 Online验收测试 - 发布 - 执行
#43 lijiahao 5c1dea0 2016-06-12 15:47:19 编译BCLOUD 29s 单元测试 6h17m 系统测试 4s 性能测试 4s 验收测试 - 执行 Online验收测试 - 发布 - 执行
#41 zhangle04 dc3e446 2016-05-25 21:49:08 编译BCLOUD 34s 单元测试 13m32s 系统测试 3s 性能测试 4s 验收测试 - 执行 Online验收测试 - 发布 - 执行
#39 lijiahao dc3e446 2016-05-13 13:25:42 编译BCLOUD 34s 单元测试 2m28s 系统测试 9s 性能测试 4s 验收测试 - 执行 Online验收测试 - 发布 - 执行
#38 zhangle04 dc3e446 2016-02-29 15:24:00 编译BCLOUD 10m6s 单元测试 - 系统测试 - 性能测试 - 验收测试 - Online验收测试 - 发布 -

数据度量和分析

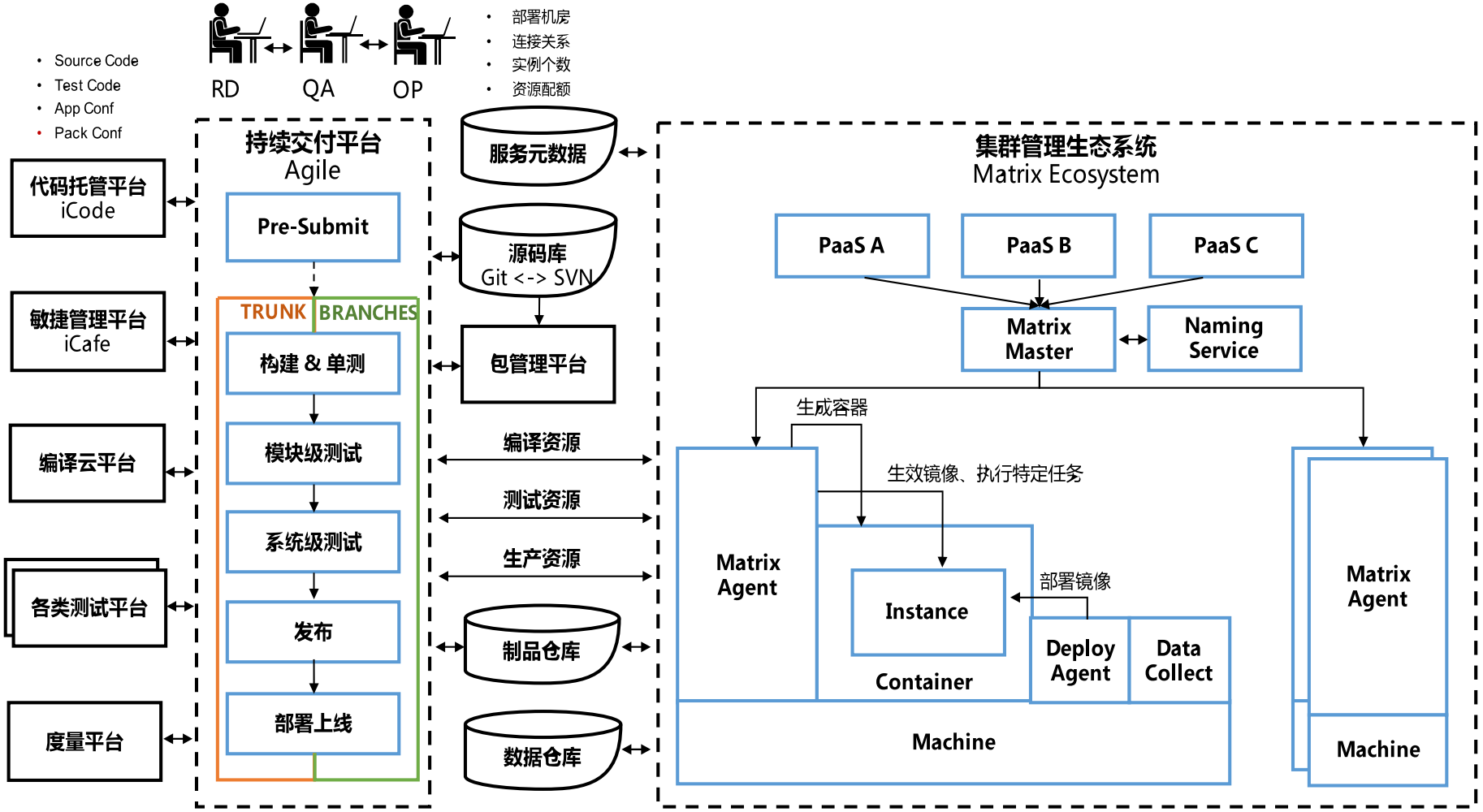


[redacted]总览成功率+构建次数表格

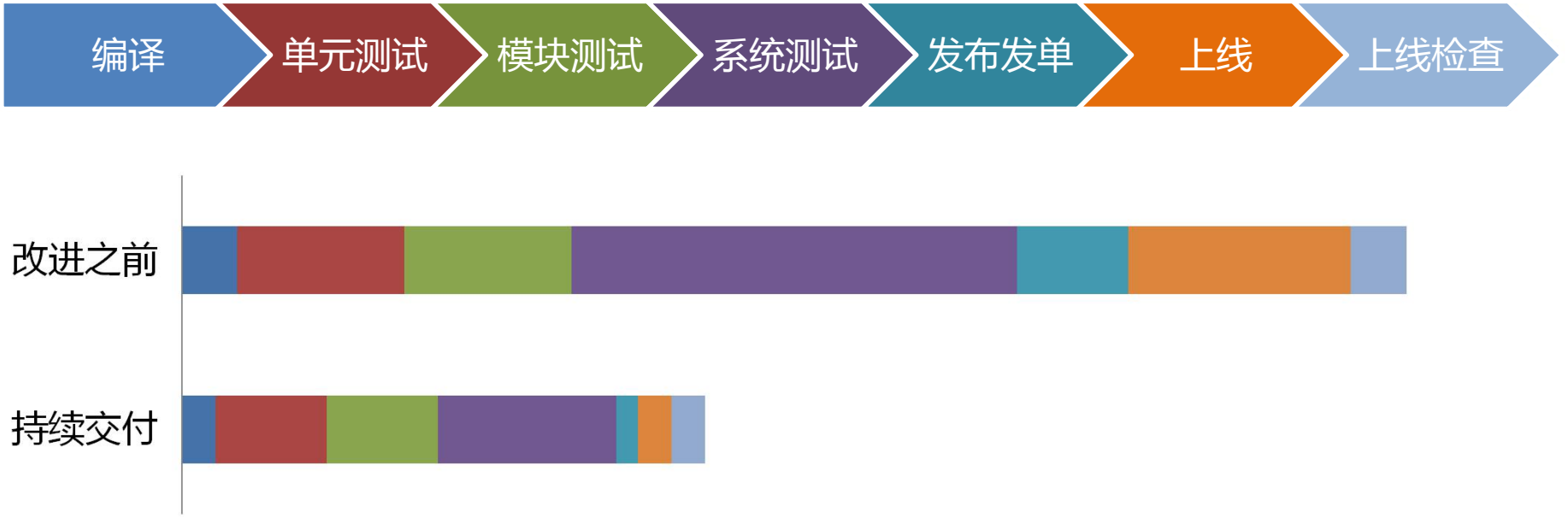
下钻顺序：团队 产品名称

	结束时间	2015-11-22		2015-11-15		2015-11-08		2015-11-01		2015-10-25	
行	团队	构建次数	构建成功率	构建次数	构建成功率	构建次数	构建成功率	构建次数	构建成功率	构建次数	构建成功率
1	推广团队	262	84.05	440	81.81	261	79.88%	138	74.05%	178	77.7%
2	试点团队	556	94.07	496	89.16	437	97.08	351	88.76	336	87.3

持续交付技术栈


RD
QA
OP
• 部署机房
• 连接关系
• 实例个数
• 资源配额
• Source Code
• Test Code
• App Conf
• Pack Conf
代码托管平台
iCode
敏捷管理平台
iCafe
编译云平台
各类测试平台
度量平台
持续交付平台
Agile
Pre-Submit
TRUNK
BRANCHES
构建 & 单测
模块级测试
系统级测试
发布
部署上线
服务元数据
源码库
Git <-> SVN
包管理平台
编译资源
测试资源
生产资源
制品仓库
数据仓库
集群管理生态系统
Matrix Ecosystem
PaaS A
PaaS B
PaaS C
Matrix
Master
Naming
Service
生成容器
生效镜像、执行特定任务
Matrix
Agent
Instance
部署镜像
Deploy
Agent
Data
Collect
Container
Machine
Matrix
Agent
Machine

持续交付效果案例回顾



- 编译、开发、测试、部署上线全面加速，整体交付周期明显缩短
- 各角色可以基于统一的交付流水线紧密协作，产品交付过程可视化、可控制
- 每日多次发布能力、故障快速回滚的能力

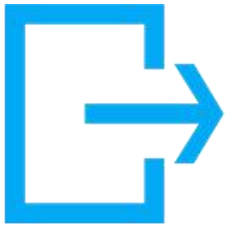
持续交付效果案例

测试

测试周期缩短69%

部署

部署耗时降低89%
每日多次发布能力
故障快速回滚能力



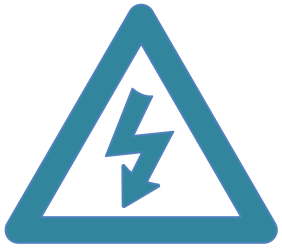
交付

开发到上线交付周期缩短53%
各角色基于交付流水线紧密协作

持续交付的价值

快速交付

从需求到交付，
端到端周期缩短



保证质量

效率提升的同时，
保证质量稳定

降低风险

稳定节奏的交付，
增强可预测性

团队做得怎么样？

Q： 仅涉及一行代码的改动需要花多长时间才能部署上线？

Q：我们是否以一种可重复且可靠的方式来做这件事的吗？

引用：《Implementing Lean Software Development》,Page. 59

Q & A

敏捷教练

Gdevops

全球敏捷运维峰会

THANK YOU！