DevOps-CICD介绍

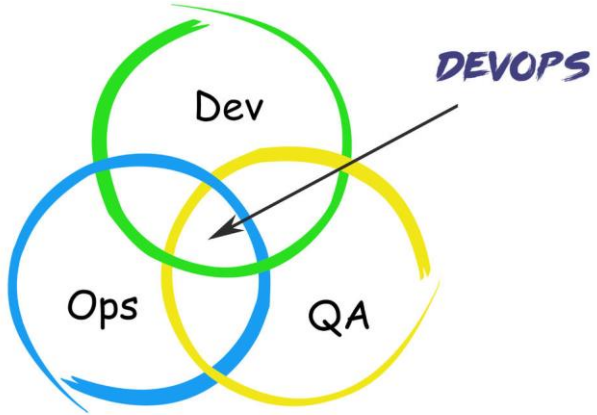
2019年7月

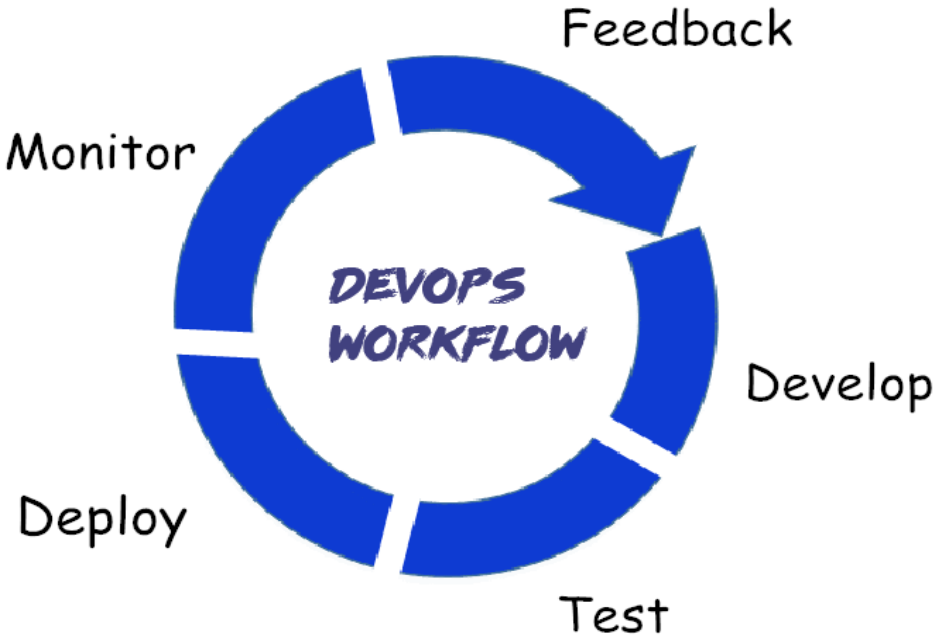
徐蕾

目录

DevOps=Culture+Automation+Measurement

(Metrics)+Sharing





communication, collaboration, and integration

Devops的概念说法不一，但大体上就是一种方法论或框架，或者是一套打破孤岛的原则。具体而言，Devops主要是关于文化，自动化，测量和共享（CAMS）

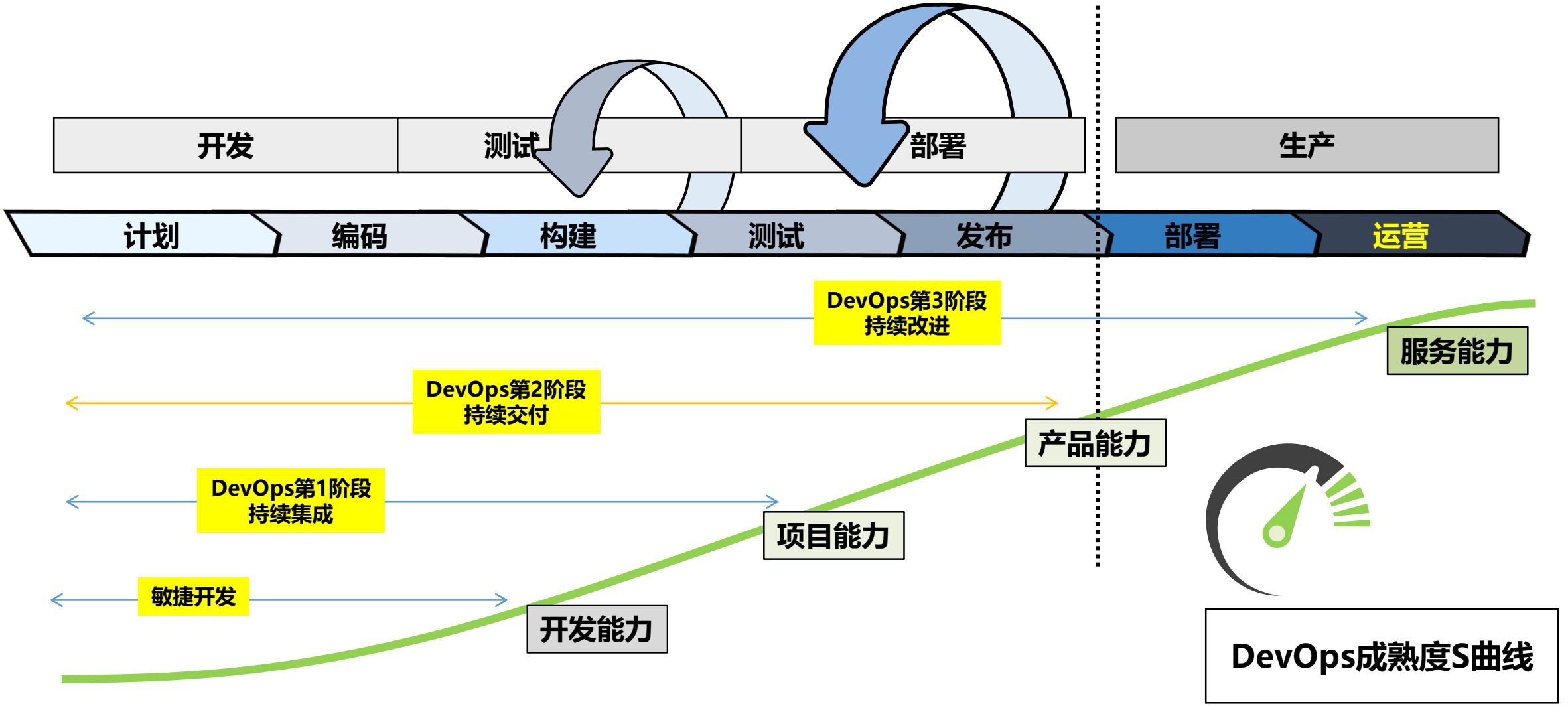
DevOps产生的原因

随着软件发布迭代的频率越来越高，传统的「瀑布型」（开发—测试—发布）模式已经不能满足快速交付的需求。DevOps是敏捷开发的延续，它将敏捷的精神延伸至IT运营（IT Operation）阶段。
敏捷开发的主要目的是响应变化，快速交付价值。

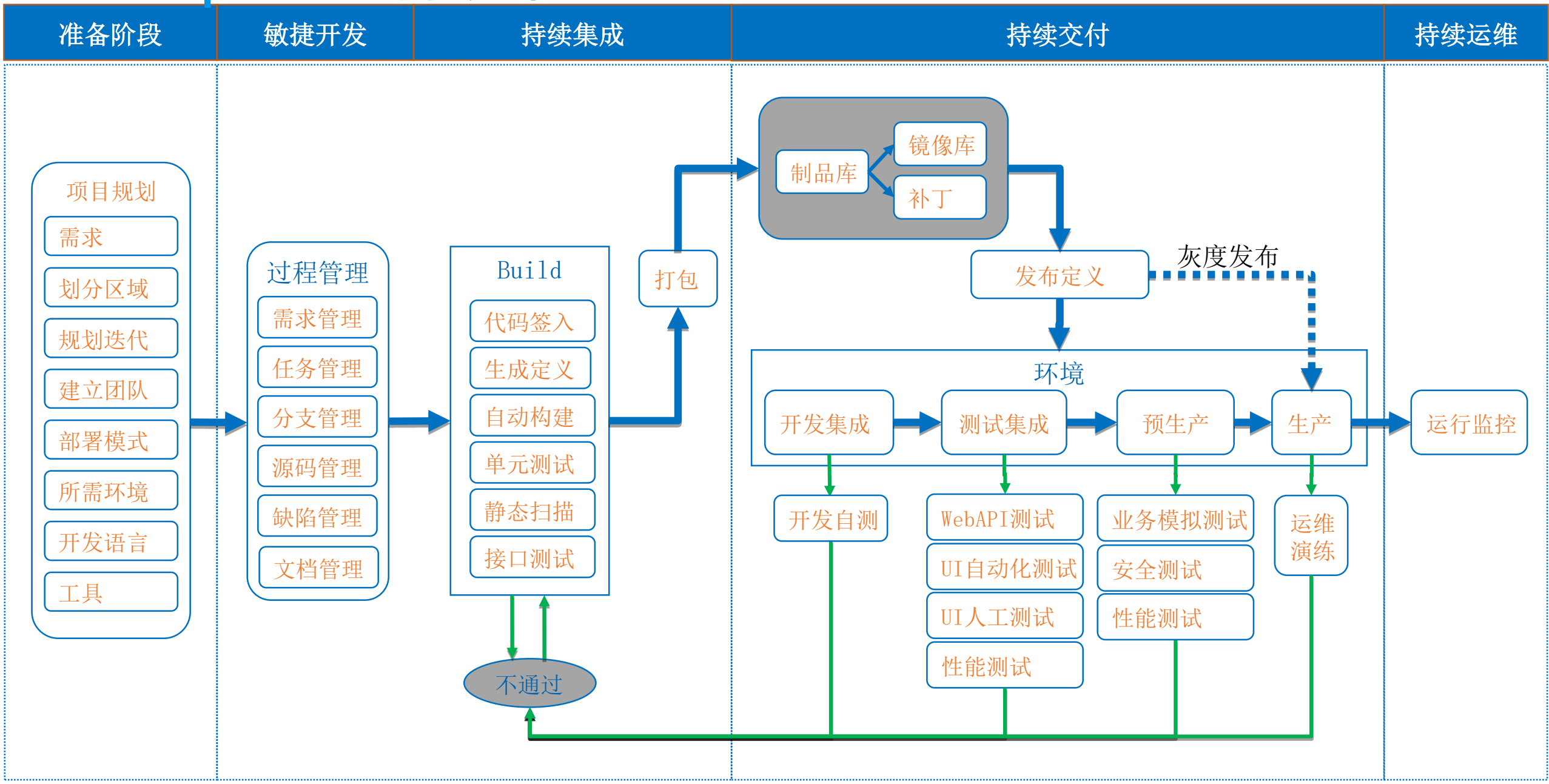
- Dev目标:Continuous Change,Add new features
- Ops目标:Continuous Stability,Create new services

敏捷驱动，贯穿整个IT流程。

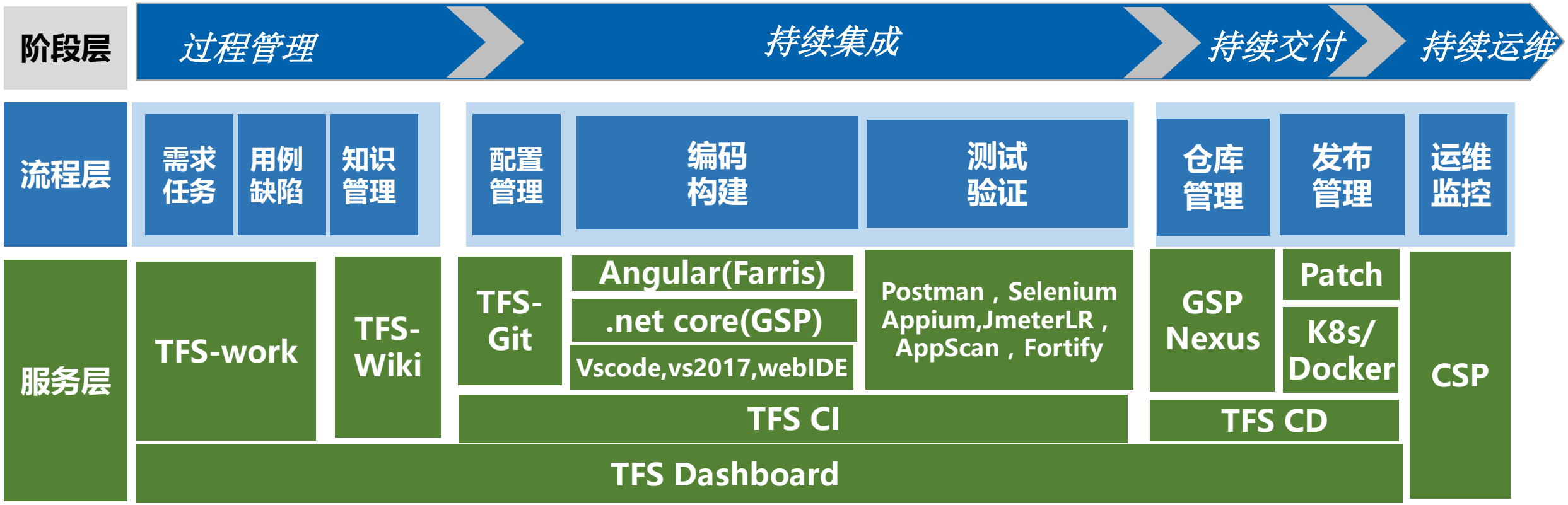
DevOps介绍--愿景



DevOps--整体流程


准备阶段
敏捷开发
持续集成
持续交付
持续运维
项目规划
需求
划分区域
规划迭代
建立团队
部署模式
所需环境
开发语言
工具
过程管理
需求管理
任务管理
分支管理
源码管理
缺陷管理
文档管理
Build
代码签入
生成定义
自动构建
单元测试
静态扫描
接口测试
不通过
打包
制品库
镜像库
补丁
发布定义
灰度发布
环境
开发集成
测试集成
预生产
生产
运行监控
开发自测
WebAPI测试
UI自动化测试
UI人工测试
性能测试
业务模拟测试
安全测试
性能测试
运维
演练

DevOps 工具链---------举例



DevOps的建设范围

能力类	DevOps体系									
能力域	敏捷开发管理			持续交付						
能力子域	需求管理	计划管理	过程管理	配置管理	构建与持续集成	测试管理	部署与发布管理	环境管理	数据管理	度量与反馈
能力项	需求收集	需求澄清与拆解	迭代管理	版本控制	构建实践	测试分级策略	部署与发布模式	环境供给方式	测试数据管理	度量指标
	需求分析	故事与任务排期	迭代活动	版本可追踪性	构建集成	代码质量管理	持续部署流水线	环境一致性	数据变更管理	度量驱动改进
	需求与用例	计划变更	过程可视化及流动			测试自动化				
	需求验收		度量分析							

目录

DevOps中核心技术-CICD

DevOps中关键的CICD

- 持续Continuous：每完成一个完整的部分，就向下个环节交付，发现问题可以马上调整。不断的获取反馈，响应反馈，降低解决问题成本。
- 集成 (Integration)：软件个人研发的部分向软件整体部分交付，以便尽早发现个人开发部分的问题；编译、测试、打包；
- 部署(deployment):代码尽快向可运行的开发/测试节交付，以便尽早测试；
- 发布(release)：具有业务影响的功能变化对最终用户可见称为“发布”。
- 交付(delivery)是指研发尽快向客户交付，以便尽早发现生产环境中存在的问题。可以理解为从 Deployment 到 Release 之间的阶段，更多的强调的是一种能力。开发有能力频繁的部署，业务有能力随时发布。

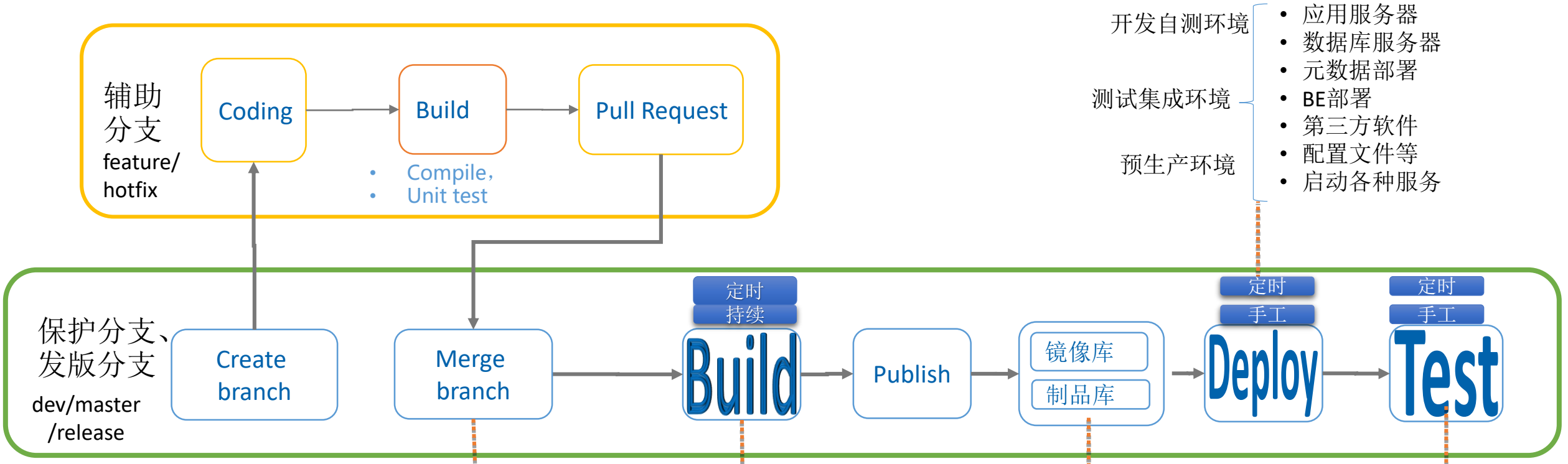
什么是持续集成？

一种软件开发实践，即团队开发成员经常集成他们的工作，通过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，从而尽早地发现集成错误。

- 持续集成是为了确保随着需求变化而变化的代码，在实现功能的同时，质量不受影响。关注代码质量。Everything is code!!!
- 基本要素：
  - 1.统一的代码库,要将所有软件资产集中放到版本库中，包括源代码，测试脚本，配置文件，部署脚本，环境描述，数据库的DDL和DML脚本,部署脚本等；
  - 2.自动构建；3.自动测试
  - 4.每个人每天都要向代码库主干提交代码（分支策略）
  - 5.每次代码递交后都会在持续集成服务器上触发一次构建
  - 6.保证快速构建（10分钟原则）
  - 7.模拟生产环境的自动测试
  - 8.每个人都可以很容易的获取最新可执行的应用程序
  - 9.每个人都清楚正在发生的状况
  - 10.自动化的部署

gscloud CICD 理想状态



质量关卡

- 人工审查
- 自动触发CICD

- 构建开发代码
- 构建10分钟原则
- 核心代码单元测试
- 静态扫描
- 单元测试覆盖率

- 版本控制
- 仓库分库规则
- 命名规范
- 权限定义
- 安全扫描

- WebApi测试100%
- 自动化UI测试
- 性能测试
- 安全测试

什么是持续交付？

- 在持续集成的基础上，将集成后的代码部署到更贴近真实运行环境的「类生产环境」（production-like environments）中。比如，完成单元测试后，可以把代码部署到连接数据库的Staging环境中更多的测试。如果代码没有问题，可以继续手动部署到生产环境中。

什么是持续部署？

持续部署：在持续交付的基础上，把部署到生产环境的过程自动化

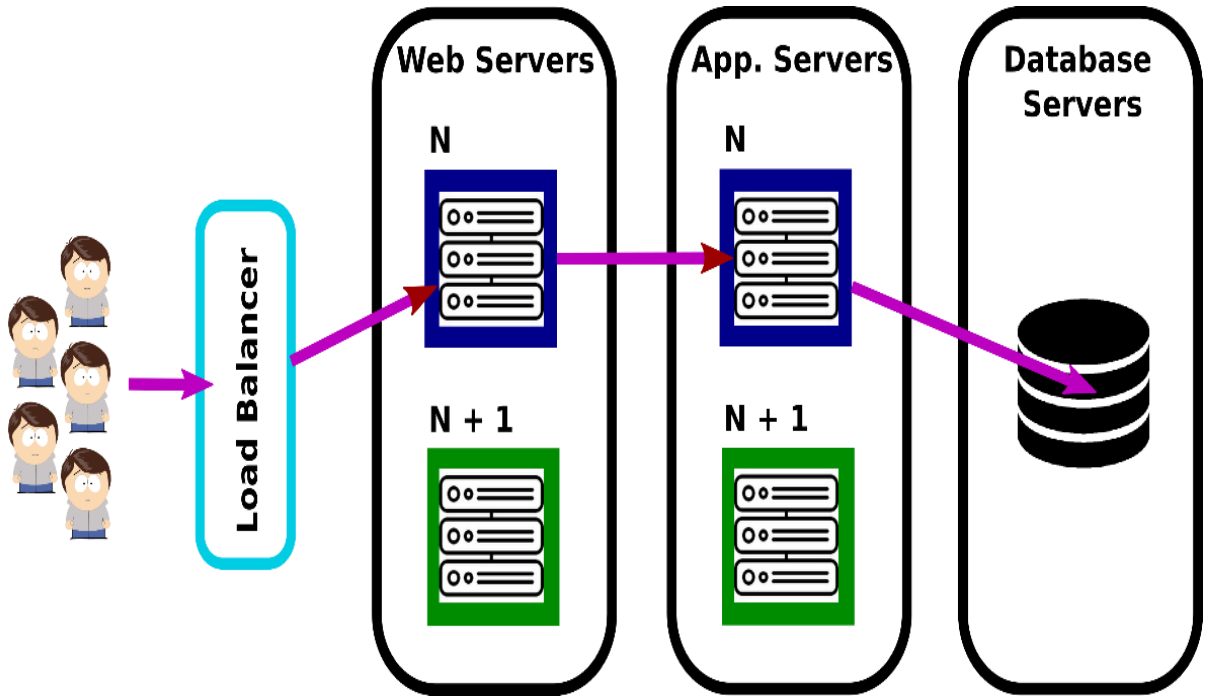
持续交付的重要点

- 1.更高频次的增量更新，这种方法有助于降低交付变更过程中涉及的成本、时间和风险。
- 2.在持续交付中找度量指标，即找出所有能让你更进一步了解用户对功能看法的度量指标，对所有这些指标进行度量。如运维环境可用性全年比例：99%，99.9%，99.99%。一年中有几天故障，一年中有几小时故障，一年中有几分钟故障。（这也就是常说的2个9,3个9,4个9）
- 3.零停机部署：
  - 功能开关（Feature Flipping）
  - 蓝 / 绿部署（Blue/Green Deployment）
  - 金丝雀发布（Canari release）---灰度发布
  - 滚动式发布

功能开关

- 功能开关可供我们在软件运行过程中启用 / 禁用相应的功能。这种技术其实非常容易理解和使用：为生产版本提供一个能彻底禁用某项功能的配置，并只在对应功能彻底完工可以正常工作后才将该属性激活。
- 举例来说，若要将某个应用程序内的一个功能全局禁用或激活：
- if Feature.isEnabled('new_awesome_feature') # Do something new, cool and awesome else # Do old, same as always stuff end
- 或者如果要真对具体用户实现类似目的：
- if Feature.isEnabled('new_awesome_feature', current_user) # Do something new, cool and awesome else # Do old, same as always stuff end

蓝绿发布



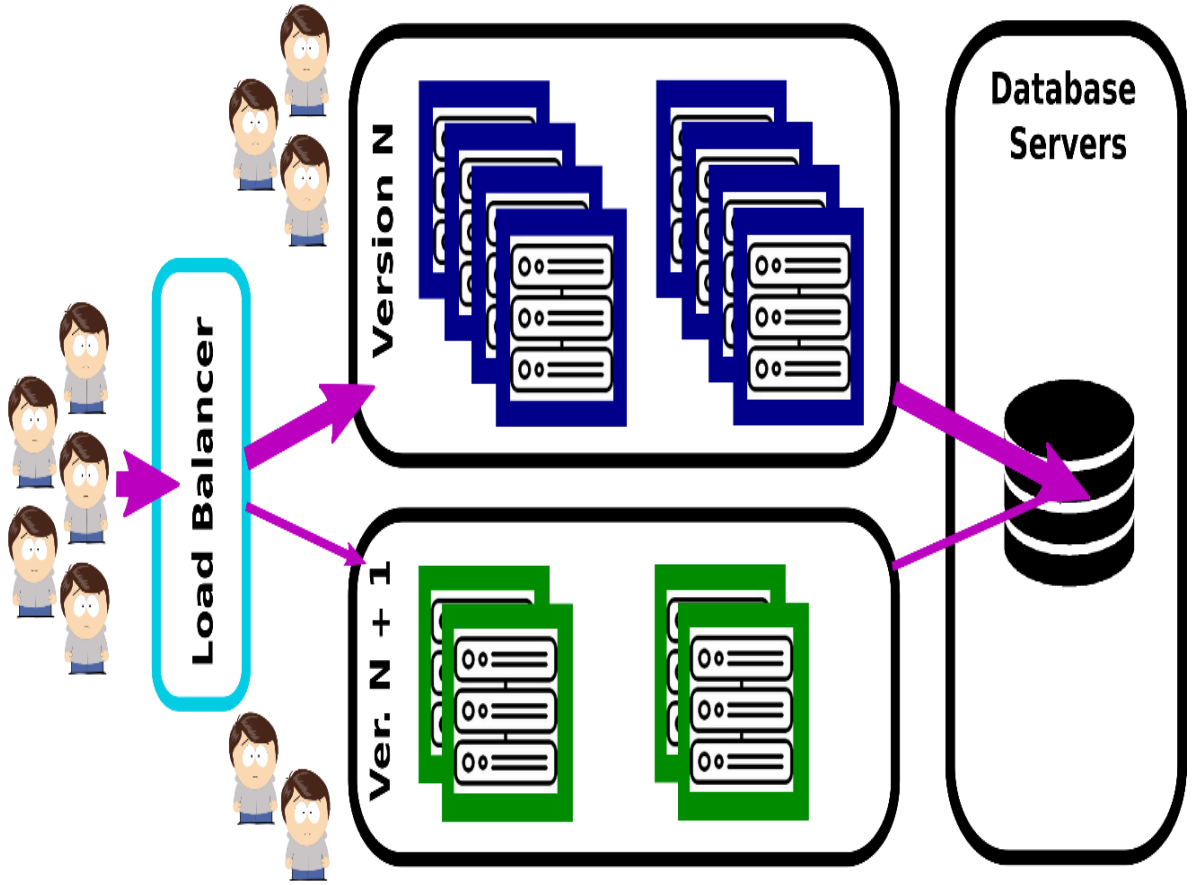
蓝 / 绿部署是指为下一版产品构建另一个完整的生产环境。开发和运维团队可以在这个单独的生产环境中放心地构建下一版产品。

当下一版产品全部完成后，可以修改负载均衡器的配置，以透明的方式将用户自动重定向至新发布的下一版。

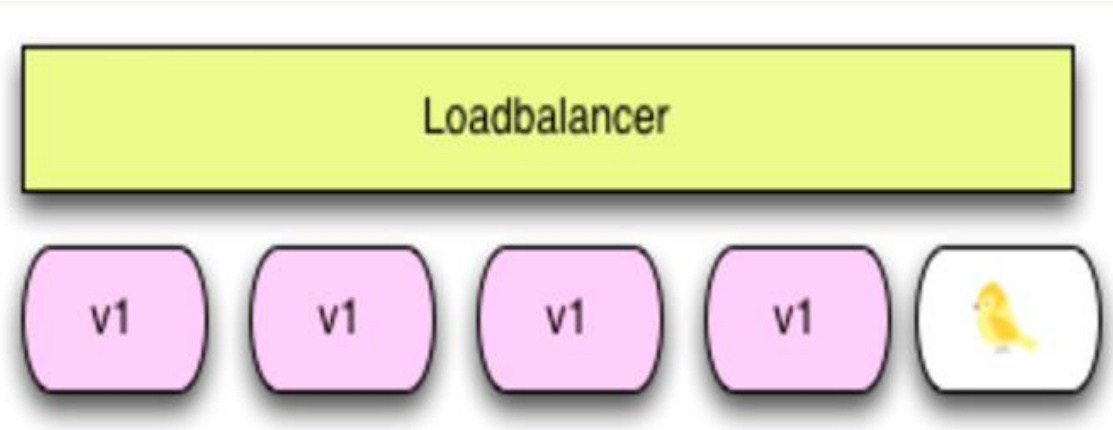
随后可将上一版的生产环境回收，用于构建下下一版的产品。

问题在于这种方式需要双倍的基础架构以及更多的服务器等。

金丝雀发布（灰度发布）



- 灰度发布是指在黑与白之间，能够平滑过渡的一种发布方式。AB test就是一种灰度发布方式，让一部分用户继续用A，一部分用户开始用B，如果用户对B没有什么反对意见，那么逐步扩大范围，把所有用户都迁移到B上面来。灰度发布可以保证整体系统的稳定，在初始灰度的时候就可以发现、调整问题，以保证其影响度，而我们平常所说的金丝雀部署也就是灰度发布的一种方式。



金丝雀发布的一般流程

- 金丝雀发布一般先发 1 台，或者一个小比例，例如 2% 的服务器，主要做流量验证用，也称为金丝雀 (Canary) 测试（国内常称灰度测试）。以前旷工开矿下矿洞前，先会放一只金丝雀进去探是否有有毒气体，看金丝雀能否活下来，金丝雀发布由此得名。简单的金丝雀测试一般通过手工测试验证，复杂的金丝雀测试需要比较完善的监控基础设施配合，通过监控指标反馈，观察金丝雀的健康状况，作为后续发布或回退的依据。
- 如果金丝测试通过，则把剩余的 V1 版本全部升级为 V2 版本。如果金丝雀测试失败，则直接回退金丝雀，发布失败。

金丝雀发布的适用场合

优势：

•用户体验影响小，金丝雀发布过程出现问题只影响少量用户

不足：

•发布自动化程度不够，发布期间可引发服务中断

适用场合：

•对新版本功能或性能缺乏足够信心

•用户体验要求较高的网站业务场景

•缺乏足够的自动化发布工具研发能力

滚动式发布

- 在金丝雀发布基础上的进一步优化改进，是一种自动化程度较高的发布方式，用户体验比较平滑，是目前成熟型技术组织所采用的主流发布方式。

- 滚动式发布一般先发 1 台，或者一个小比例，如 2% 服务器，主要做流量验证用，类似金丝雀 (Canary) 测试。
- 滚动式发布需要比较复杂的发布工具和智能 LB，支持平滑的版本替换和流量拉入拉出。

滚动式发布的一般流程

- 每次发布时，先将老版本 V1 流量从 LB 上摘除，然后清除老版本，发新版本 V2，再将 LB 流量接入新版本。这样可以尽量保证用户体验不受影响。
- 一次滚动式发布一般由若干个发布批次组成，每批的数量一般是可以配置的（可以通过发布模板定义）。例如第一批 1 台（金丝雀），第二批 10%，第三批 50%，第四批 100%。每个批次之间留观察间隔，通过手工验证或监控反馈确保没有问题再发下一批次，所以总体上滚动式发布过程是比较缓慢的 (其中金丝雀的时间一般会比后续批次更长，比如金丝雀 10 分钟，后续间隔 2 分钟)。
- 回退是发布的逆过程，将新版本流量从 LB 上摘除，清除新版本，发老版本，再将 LB 流量接入老版本。和发布过程一样，回退过程一般也比较慢的。

滚动式发布的适用场合

- **优势：**
- 用户体验影响小，体验较平滑
- **不足：**
- 发布和回退时间比较缓慢
- 发布工具比较复杂，LB 需要平滑的流量摘除和拉入能力
- **适用场合：**
- 用户体验不能中断的网站业务场景
- 有一定的复杂发布工具研发能力；

目录

谢谢！