中南大学
CENTRAL SOUTH UNIVERSITY

软件工程

Software Engineering

龙 军

jlong@csu.edu.cn 18673197878

计算机学院 | 大数据研究院

第十章 软件运维

大 纲

01-软件运营

02-软件维护

03-开发运维一体化DevOps

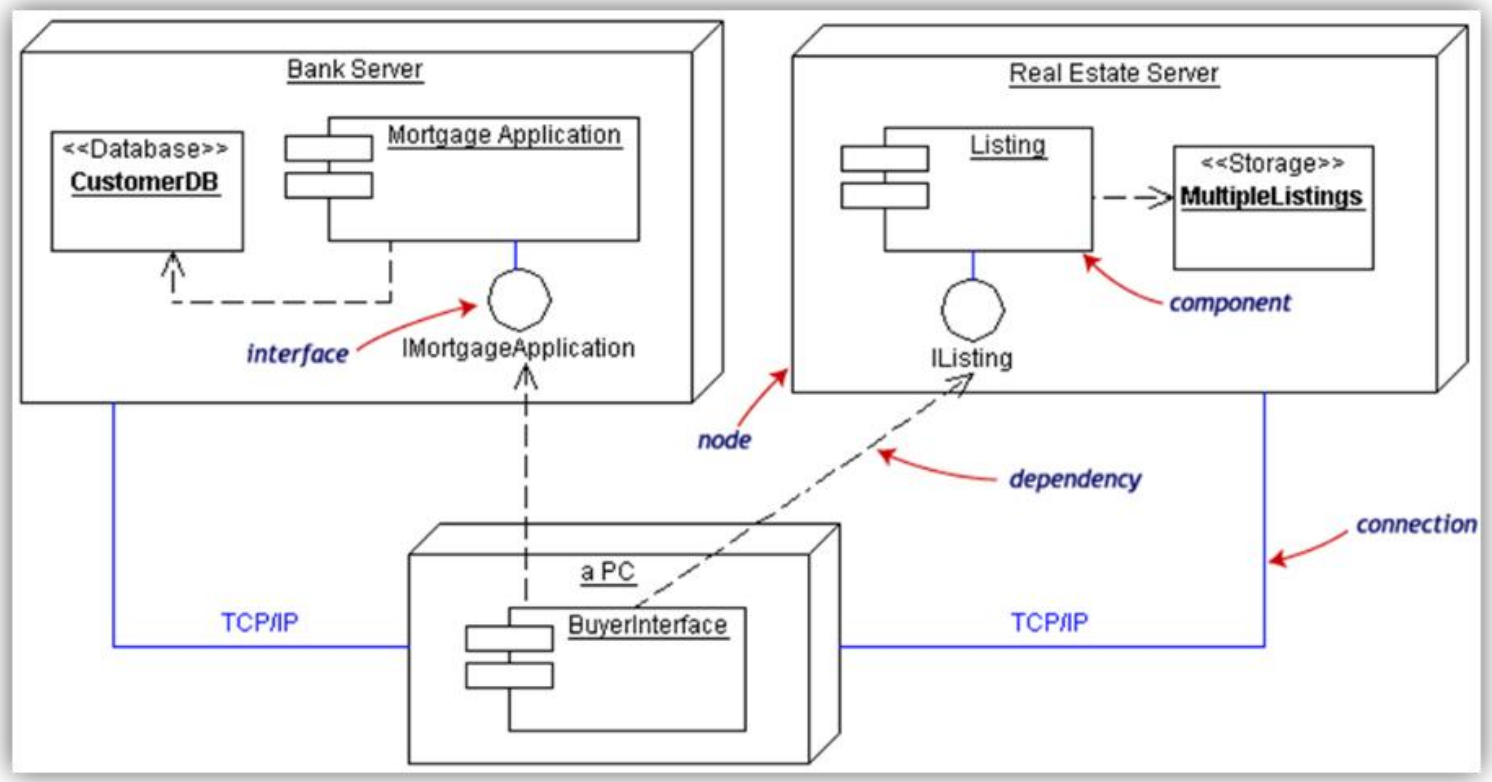
@第9章和第2.4节.教材

软件运营 (Operation)

- 软件运营，是指在目标环境中**部署**和**配置**软件，以及在软件运行时（直到停用）对其进行**监控**和**管理**的过程
- 目标
  - 快速部署
  - 配置优化
  - 持续健康运行

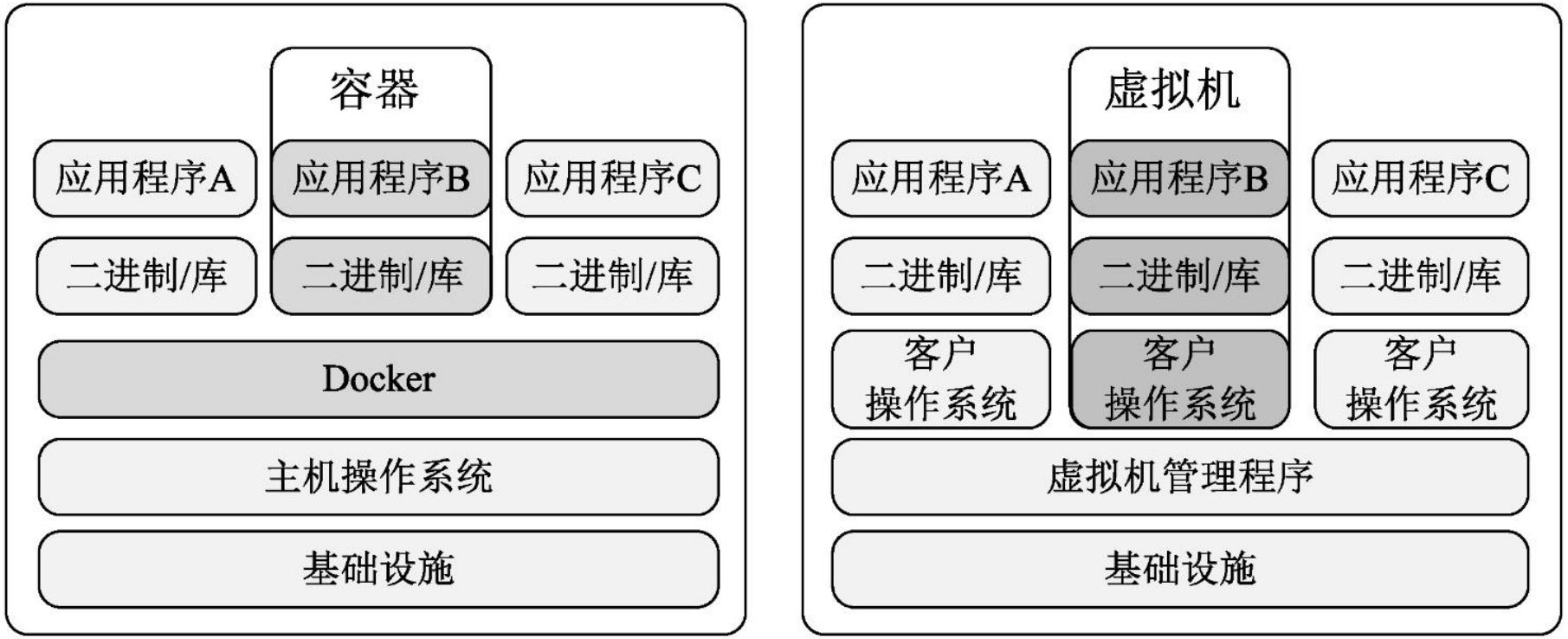
软件部署

- 根据软件架构的部署视图进行打包、安装和发布



软件部署的推荐方法

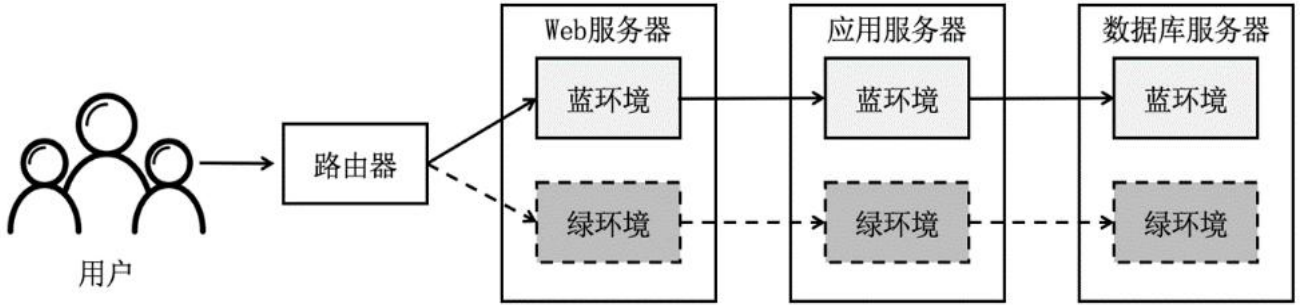
- 基于容器或虚拟机进行应用镜像的打包与安装
  - 容器是轻量级的虚拟化技术，如docker



软件部署的策略

- 停机部署（big bang deployment）
- 蓝绿部署（blue-green deployment）

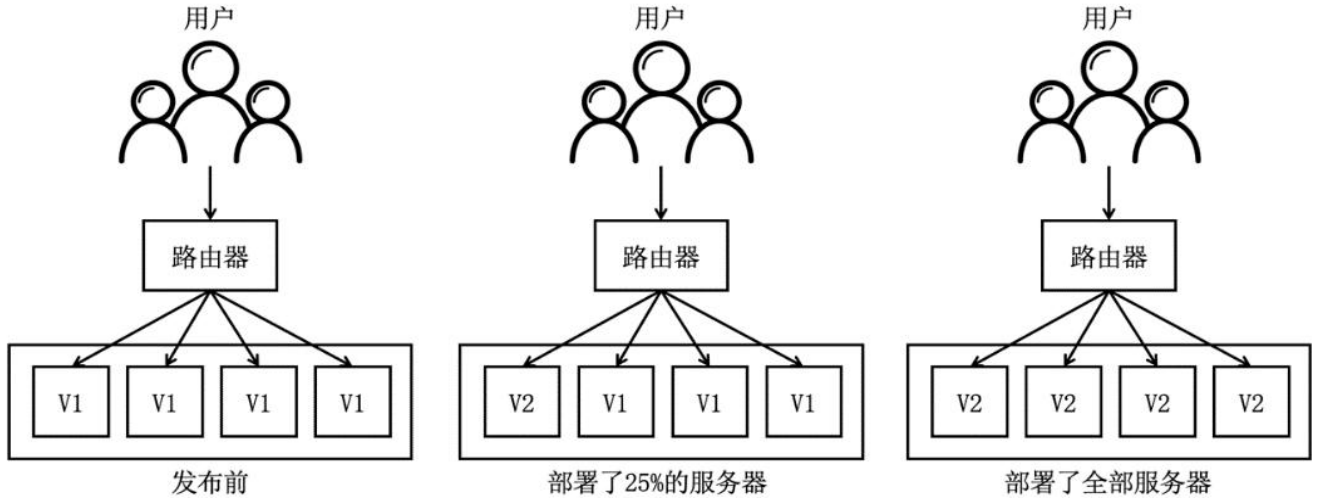
在生产线上部署相同数量的新版本服务。旧版本的生产环境称为蓝环境，用于对外提供软件服务；新版本的预发布环境称为绿环境，用于对新版本进行测试。



- 灰度部署（canary deployment）

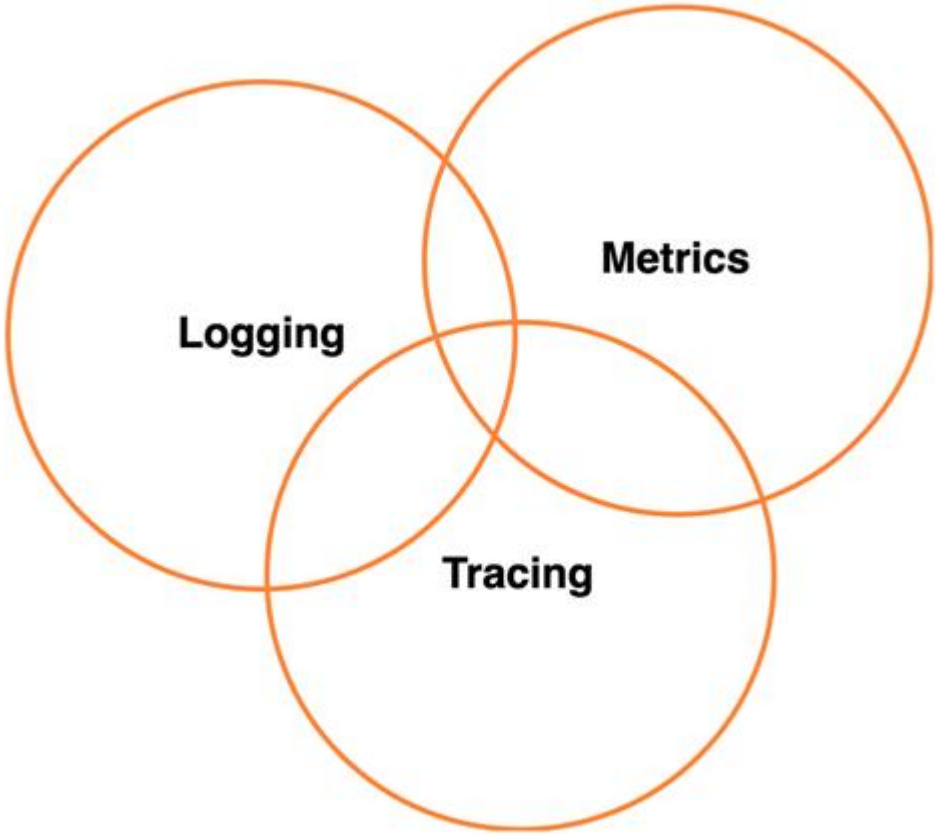
又称为金丝雀部署

新版本进行增量部署，先部署一部分，通过路由器配置将用户流量引流至新版本部署中；如果没有问题，再部署和引流一部分，直到全部部署和引流完成



软件运营监控

- 搭建**可观测性**平台
  - **Logging**: 离散的日志信息
  - **Metrics**: 聚合的度量指标
  - **Tracing**: 请求级别的链路追踪



- 技术栈举例
  - Spring Boot Actuator 进行健康检查
  - Micrometer 收集业务指标
  - Prometheus 进行指标的存储和查询
  - Grafana 进行可视化展示

大 纲

中南大學
CENTRAL SOUTH UNIVERSITY

第十章 软件运维

01-软件运营

02-软件维护

03-开发运维一体化DevOps

@第9章和第2.4节.教材

软件维护

- 软件**上线运行**后的修改称为软件维护
- 软件需要持续维护：
  - **出现故障**，需要纠正
    - 潜在的缺陷产生软件错误，需纠正
  - **需求变化**，需要升级
    - 软件需求发生变化，需要增强功能
  - **环境变化**，需要适应
    - 需要改变软件以在新的环境中运行
  - **系统脆弱**，需要改善
    - 软件在持续维护过程中变得脆弱，需要改善软件设计

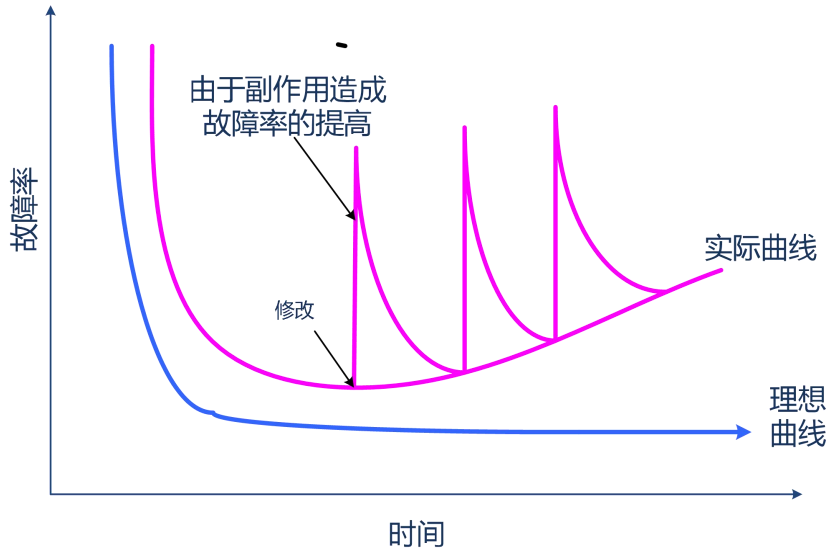
软件维护 “冰山”

- 有关调查结果表明，软件维护是软件工程中最消耗资源的活动，很多软件公司中软件维护的成本已经达到了整个软件生存周期资源的75%，甚至达到了90%。
- 挑战：
  - 软件系统趋于大型化和复杂化
  - 大多数软件在设计时没有考虑到将来的修改
  - 开发人员变动频繁
  - 文档不够详细
  - 维护周期长
  - 维护与使用同步进行
- 后果：导致维护活动的时间与花费不断增加。

软件行业的演变：从制造产业到服务产业

- 软件工程伊始，软件开发被认为是制造业，但是，难以用销售价值决定软件价值，因为 75% 的资源都消耗在软件维护上。

- 因为软件维护的开销巨大，软件产业逐渐被认为是服务产业。



维护类型

- 纠错性维护（corrective maintenance）
  - 为修复所发现的问题而进行的反应式维护
- 预防性维护（preventive maintenance）
  - 在软件产品中的潜在错误成为实际错误前所进行的预防性修复，例如设计重构和代码优化等
- 适应性维护（adaptive maintenance）
  - 为适应环境的变化而修改软件的维护
- 补充性维护（additive maintenance）
  - 添加新功能或新特性以增强产品使用的维护
- 完善性维护（perfective maintenance）
  - 为用户提供功能增强、程序文档改进、软件性能和可维护性等质量属性的提升而做的维护。和补充性维护不同的是，它并不增加新功能和新特性，所做的修改较小
- 紧急维护（emergency maintenance）
  - 计划外的应急修改，以暂时保持系统运行，等待后续纠错性维护

维护的技术问题

- 有限理解（Limited understanding），对他人开发软件进行维护时，如何快速理解程序并找到需要修改或纠错的地方？
- 在软件维护过程中需要大量的回归测试，耗时耗力。
- 当软件非常关键以致不能停机时，如何进行在线维护而不影响软件的运行？
- 影响分析，如何对现有软件的变更进行全面影响分析？
- 如何在开发中促进和遵循软件的可维护性？

维护成本

- 维护的工作可划分成：
  - **生产性活动** 如，分析评价、修改设计、编写程序代码等
  - **非生产性活动** 如，程序代码功能理解、数据结构解释、接口特点和性能界限分析等
- 维护工作量的模型

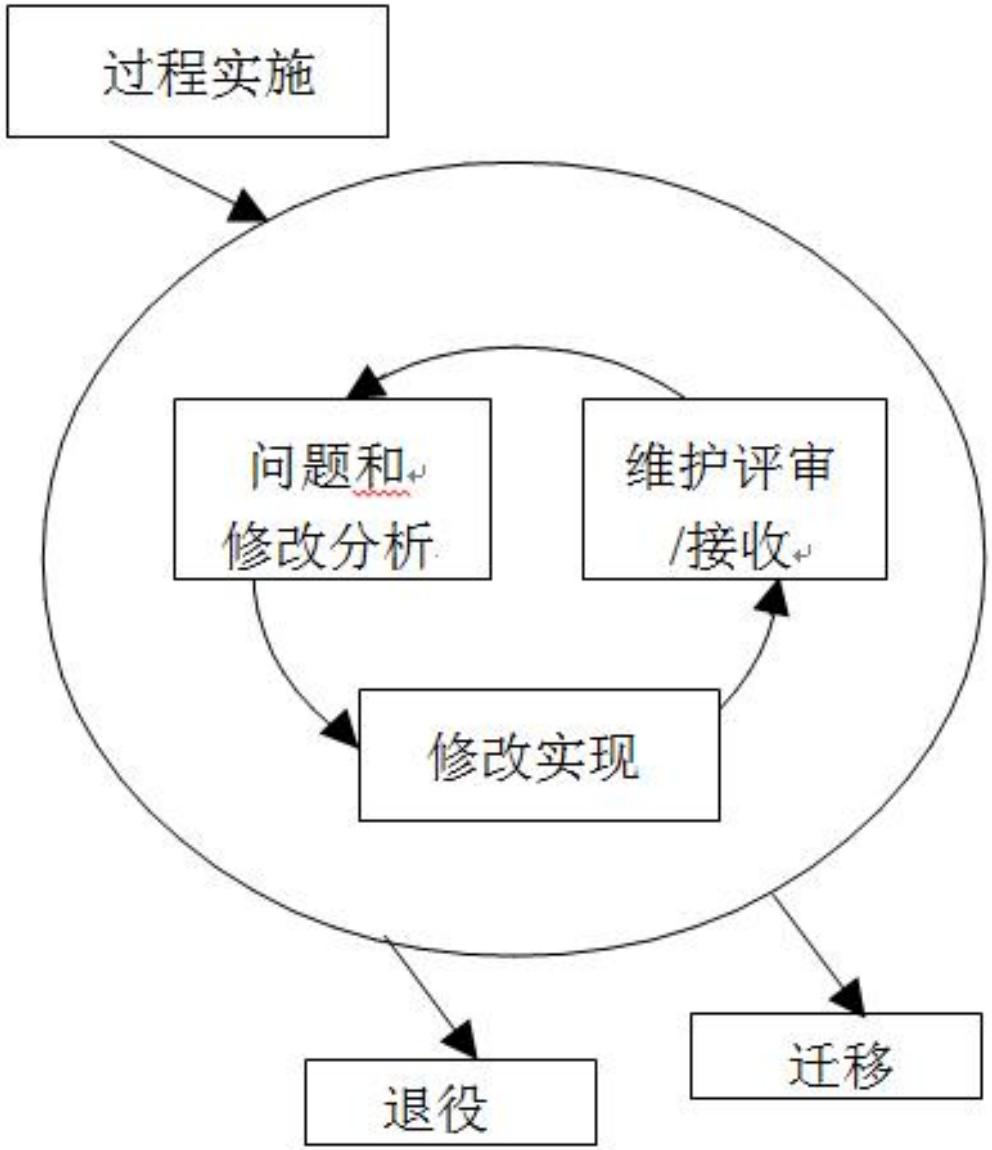
$$M = p + Ke^{c-d}$$

  - M：维护的总工作量；
  - P：生产性工作量；
  - K：经验常数；
  - e：软件的规模；
  - c：复杂程度；
  - d：维护人员对软件的熟悉程度

影响维护成本的因素

- 操作环境：硬件和软件
  - 软件的规模越大、复杂性越高、年龄越大，硬件的能力越低，软件维护的成本和工作量就越大。
- 组织环境：策略、竞争力、过程、产品和个人
  - 软件开发过程和维护过程越规范，采用的设计方法和编程语言模块化程度越高，工程师对软件的熟悉程度越高、能力越强，产品的可靠性和安全性要求越低，软件维护的成本和工作量就越小。

维护过程



维护活动

- **过程实施**（Process Implementation）
  - 建立维护过程期间应执行的计划和规程，包括维护计划、维护规程、问题解决规程、用户反馈计划、移交计划、配置管理计划。
- **问题和修改分析**（Problem and Modification Analysis）
  - 在修改软件前，要分析修改请求/问题报告，以确定其对组织、现行系统和接口系统的影响，提出可能的方案建议并形成文档，通过核准形成期望的解决方案。
- **修改实现**（Modification Implementation）
  - 根据计划和方案更新相应的需求、设计和代码，并进行测试等软件验证工作。

- **维护评审/接收**（Maintenance Review/Acceptance）
  - 对上述的维护进行评审，以确保对软件的修改是正确的，并且这些修改是使用正确的方法按批准的要求完成的。
- **迁移**（Migration）
  - 在软件的生存周期期间，如果需要将它迁移到一个新环境，则应制订迁移计划、通告用户迁移、提供迁移培训、把软件迁移至新环境、通告迁移完成情况、评估新环境的影响、并进行旧软件和数据的归档。在迁移实施时，旧环境和新环境可以并行进行工作，以便平稳迁移到新环境。
- **退役**（Retirement）
  - 软件一旦结束使用生存周期，必须退役。退役时，要制定退役计划、通知用户退役、提供退役培训、实施退役、通告退役完成情况、并进行旧软件和数据的归档。在制定退役计划时，要分析退役的成本和影响、决定是局部还是全部退役、是否用新软件来代替退役软件等。

软件维护技术

- 程序理解
- 逆向工程（Reverse Engineering）
- 再工程（Reengineering）

程序理解

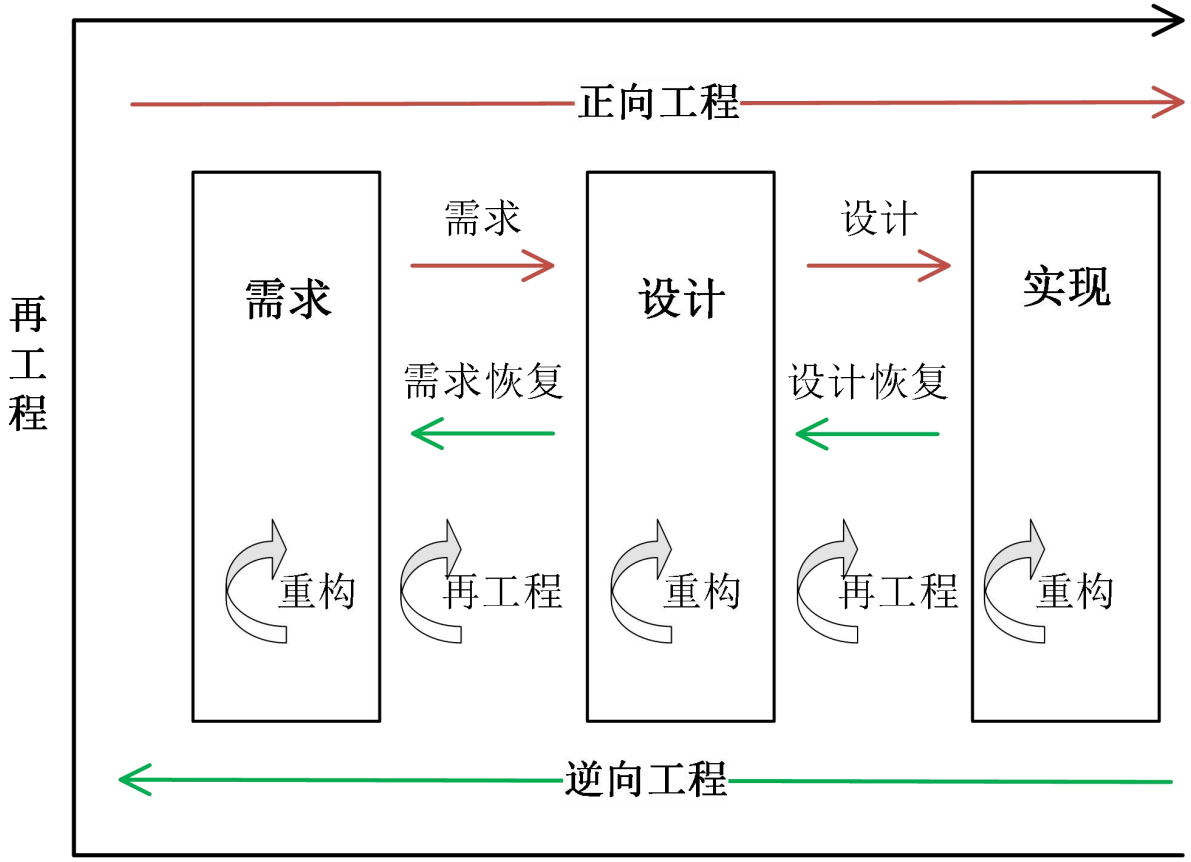
- 软件维护的总工作量大约一半被用在理解程序上。
- 程序理解通过提取并分析程序中各种实体之间的关系，形成系统的不同形式和层次的抽象表示，完成程序设计领域到应用领域的映射。
- 程序员在理解程序的过程中，经常通过反复三个活动——阅读关于程序的**文档**，阅读**源代码**，**运行程序**来捕捉信息。
- 程序理解工具
  - 基于程序结构的**可视化**工具，通过分析程序的结构，抽取其中各种实体，使用图形表示这些实体和它们之间的关系，可以直观地为维护者提供不同抽象层次上的信息。
  - 帮助维护者**导航浏览**源代码，为浏览工作提供着眼点，缩小需要浏览的代码范围。
  - 基于**大模型**的代码自动问答和代码检索。

逆向工程

- 逆向工程是分析软件，识别出软件的组成成份及其相互的关系，以更高的抽象层次来刻画软件的过程，它并不改变软件本身，也不产生新的软件。
- 逆向工程主要分为以下几类:
  - 重新文档化（redocumentation）：分析软件，改进或提供软件新的文档；
  - 设计恢复（design recovery）：从代码中抽象出设计信息；
  - 规约恢复（specification recovery）：分析软件，导出需求规约信息；
  - 重构（refactoring, restructuring）：在同一抽象级别上转换软件描述形式，而不改变原有软件的功能；
  - 数据逆向工程（data reverse engineering）：从数据库物理模式中获取逻辑模式，如实体关系图。

再工程

- 再工程是在逆向工程所获信息的基础上修改或重构已有的软件，产生一个新版本的过程，它将逆向工程、重构和正向工程组合起来，将现存系统重新构造为新的形式。



讨论

- 一个大型大学有一个大型计算机系统，用于存储和管理所有学生和教职工的信息。该系统：已经使用了25年，它采用Cobol结构化程序设计技术开发，并与关系数据库通信；它运行在一台IBM主机上；有50多万行代码。该系统已经进行过多次修改，既有经过策划的修改，也有快速修改，现在维护的成本过高。认识到有这些问题，大学希望利用面向对象的开发优势，但是不幸的是，维护这个系统的90%以上的员工都是新人，并不熟悉系统的实现。
- 如何办?

大 纲

第十章 软件运维

01-软件运营

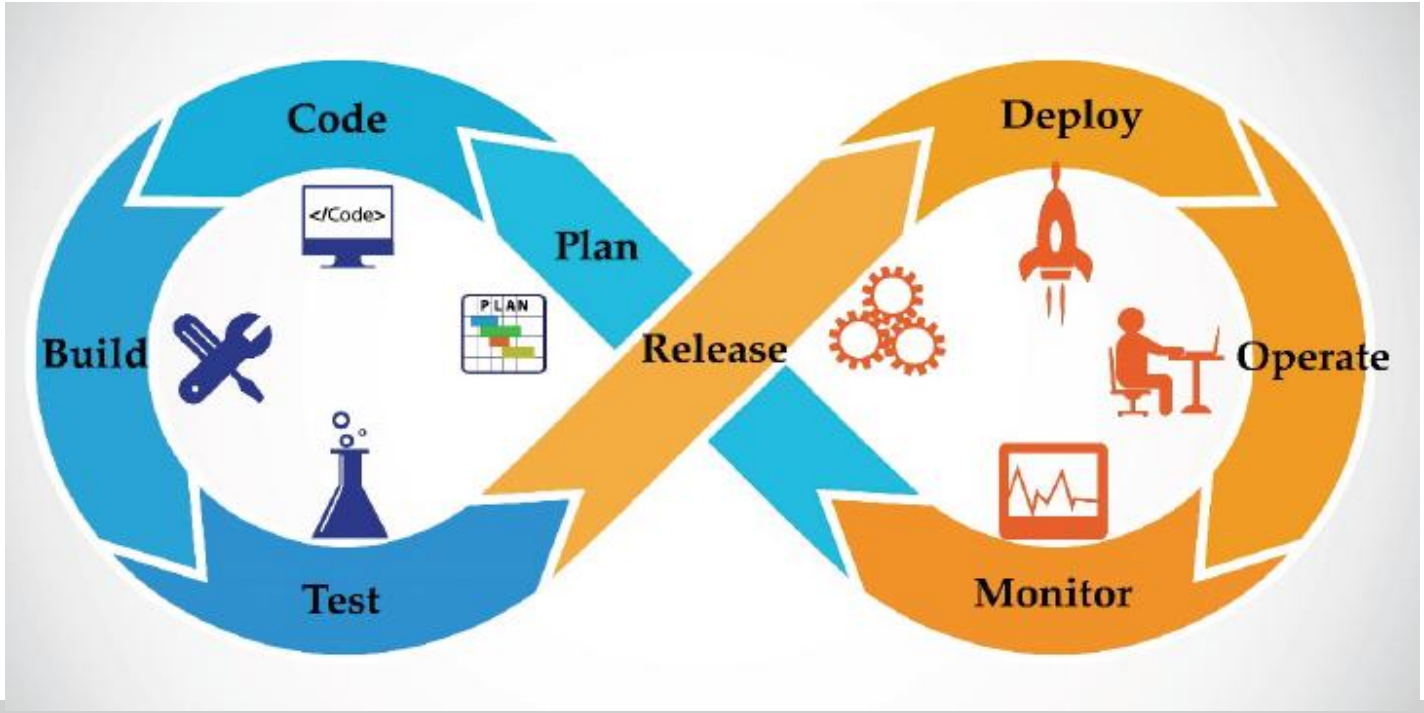
02-软件维护

03-开发运维一体化DevOps

@第9章和第2.4节.教材

DevOps

- DevOps（Development & Operations）开发运维一体化
  - （持续交付）快速实现一行代码的变更，到软件交付到用户手中
  - （自动化）从代码提交到最终交付用户只需要按下按钮，自动化每一个工作环节，及时收到用户反馈



背景

Flickr一天内完成十几次的部署
Etsy每天50次以上的部署
Amazon目前每周部署1000次

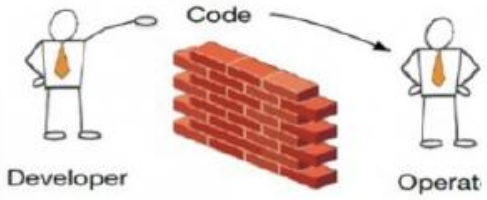
软件交付从产品转变为服务，发布周期越来越短，频率越来越高

Facebook一天内上千次配置参数更改，在90天内28%的配置被创建或更新

需要开发运维一体化！


TECHNOLOGY LAB / INFORMATION TECHNOLOGY

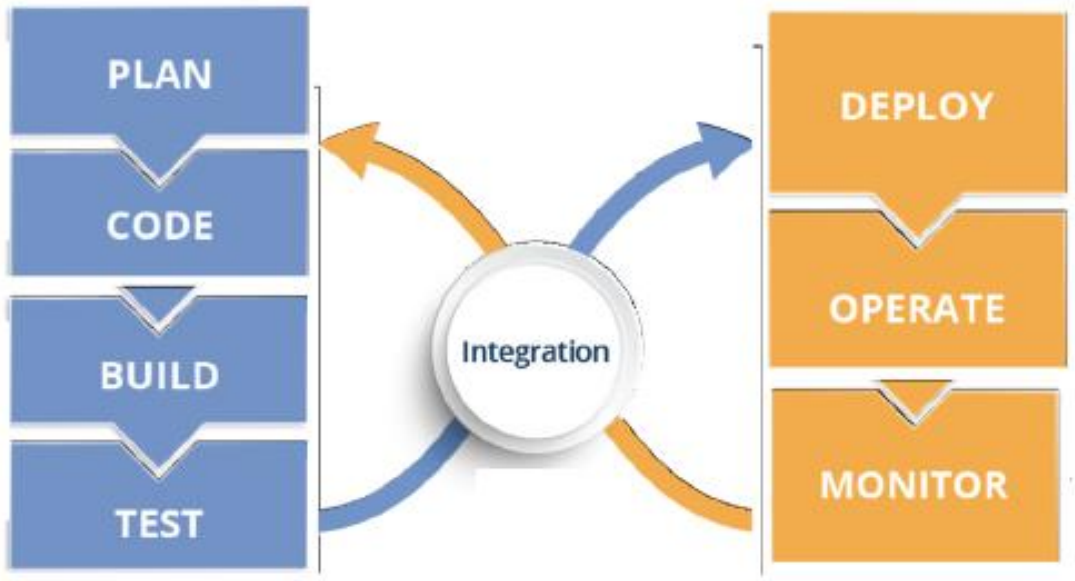
Why Gmail went down: Google misconfigured load balancing servers (Updated)




Amazon Outage Crashes Reddit, Quora, and Other Websites

“开发部署—运行维护”阶段割裂

DevOps过程



Plan：项目规划、设计等

Code：代码开发和审阅，版本控制等

Build：项目构建、打包

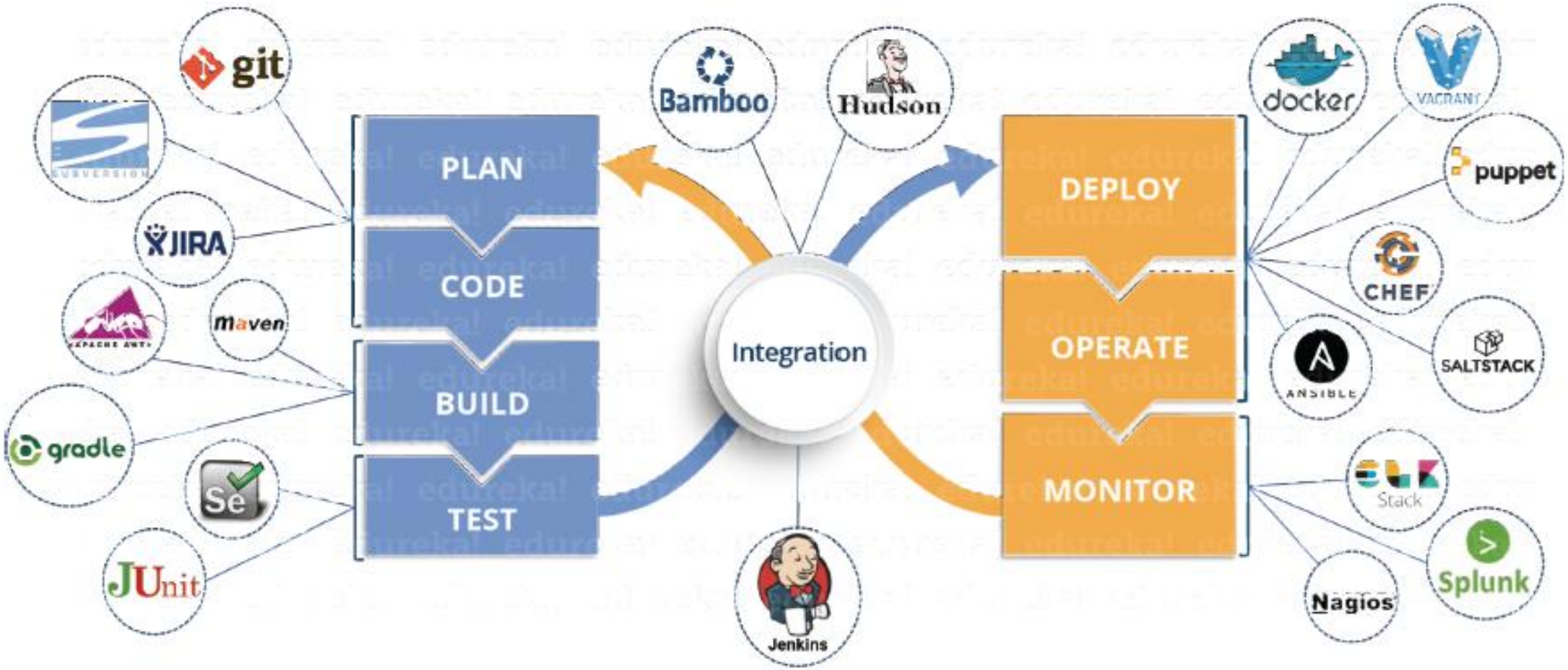
Test：单元测试、集成测试、负载测试等

Deploy：应用的配置与部署

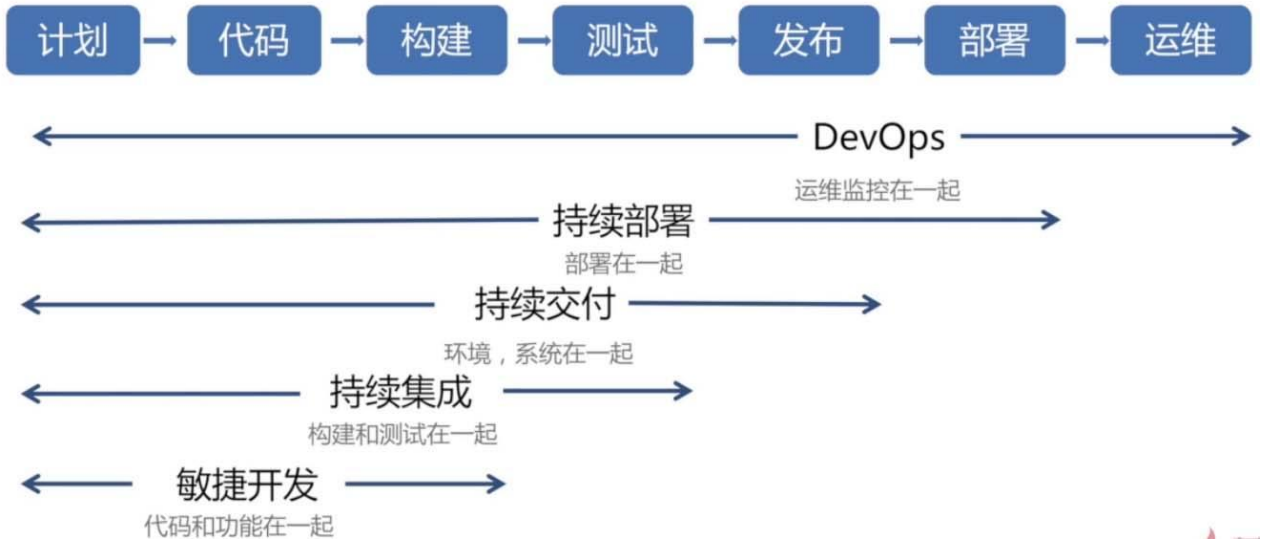
Operate：应用的再配置、再部署、升级等

Monitor：应用运行时监控、最终用户体验

DevOps工具集

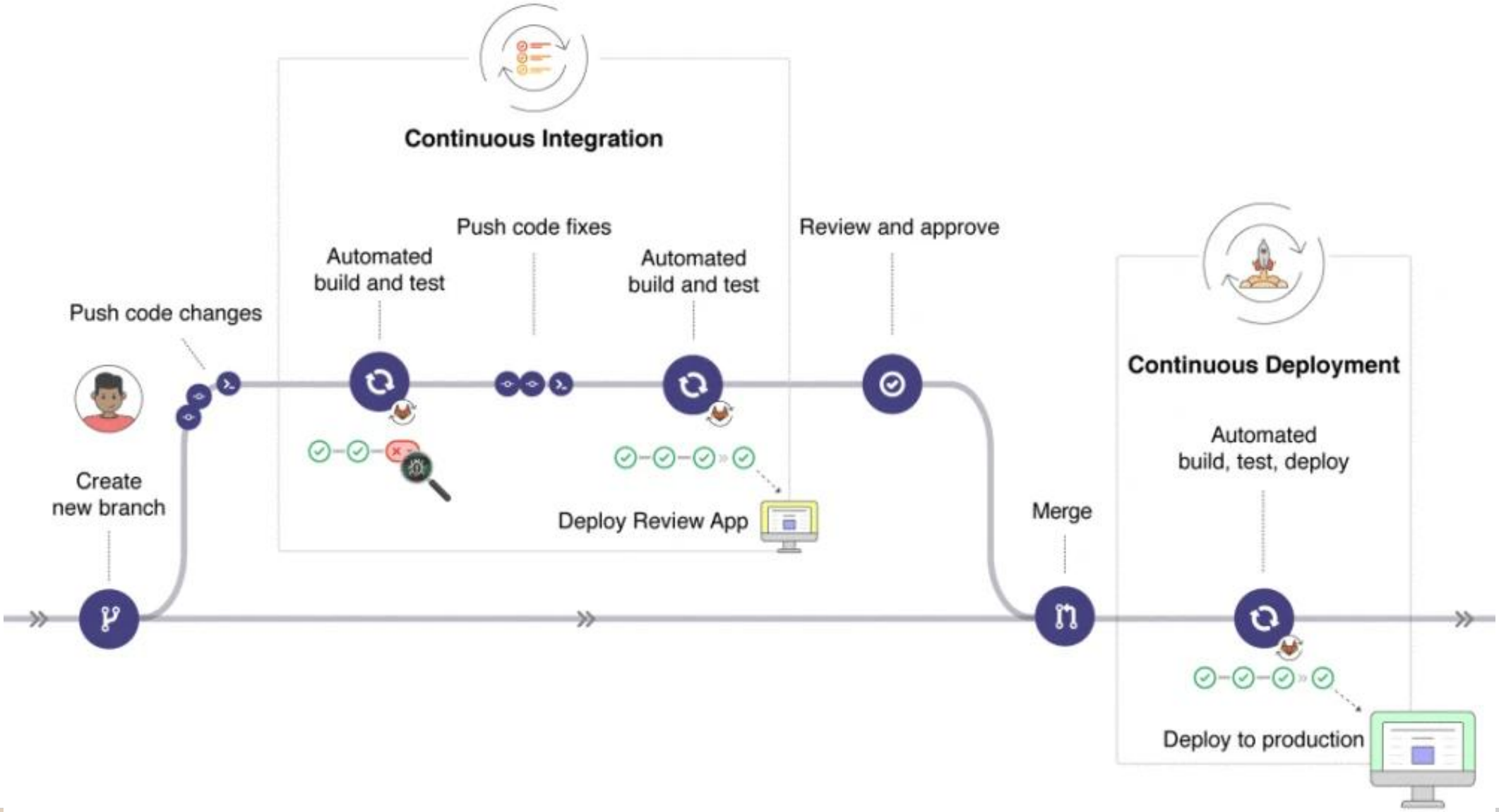

git
SUBVERSION
JIRA
APACHE ANT
maven
gradle
Se
JUnit
PLAN
CODE
BUILD
TEST
Bamboo
Hudson
Integration
Jenkins
DEPLOY
OPERATE
MONITOR
docker
VAGRANT
puppet
CHEF
ANSIBLE
SALTSTACK
Stack
Splunk
Nagios

持续集成、持续交付和持续部署 (CICD)



- DevOps 的根本理念是 “持续”，最核心的技术实践是持续集成、持续交付和持续部署。
- 持续集成（continuous integration）：开发人员频繁地（一天多次）将更新的代码合并或者提交到主干源码仓库中。这伴随着执行一系列的质量保证活动如代码规范检查、单元测试、安全扫描、人工评审等来确保代码的质量。
- 持续交付（continuous delivery）：在持续集成的基础上，将集成后的代码自动安装到更贴近真实运行环境的 “类生产环境/预生产环境” 中。
- 持续部署（continuous deployment）：在持续交付的基础上，将交付后的代码自动部署到生产环境中。

Git中的CICD


Continuous Integration
Push code fixes
Review and approve
Automated
build and test
Automated
build and test
Push code changes
Create
new branch
Deploy Review App
Merge
Continuous Deployment
Automated
build, test, deploy
Deploy to production