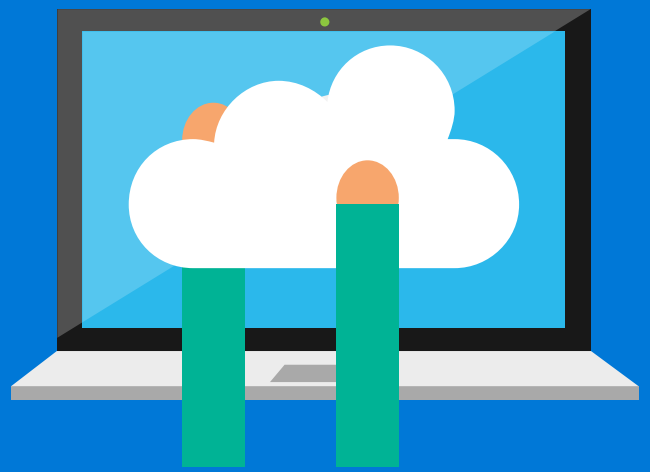
【DevOps云端开发训练营】

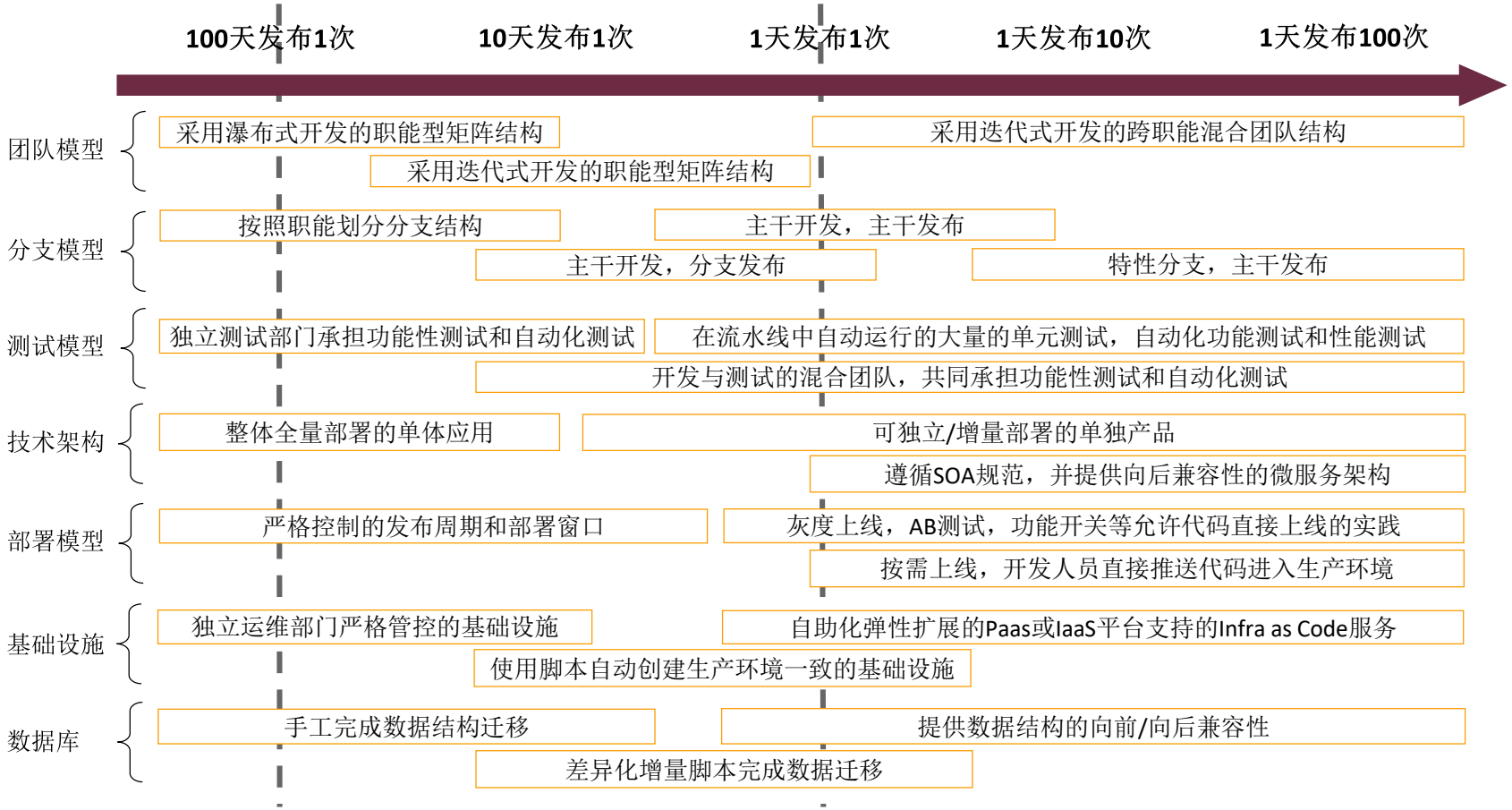
2.2 – 持续交付与DevOps流水线

Agenda



- 100-to-100 持续交付实施框架
- 持续交付就是持续解耦
- 编码工作流
- 测试工作流
- 发布工作流

持续交付实施框架



容器的价值

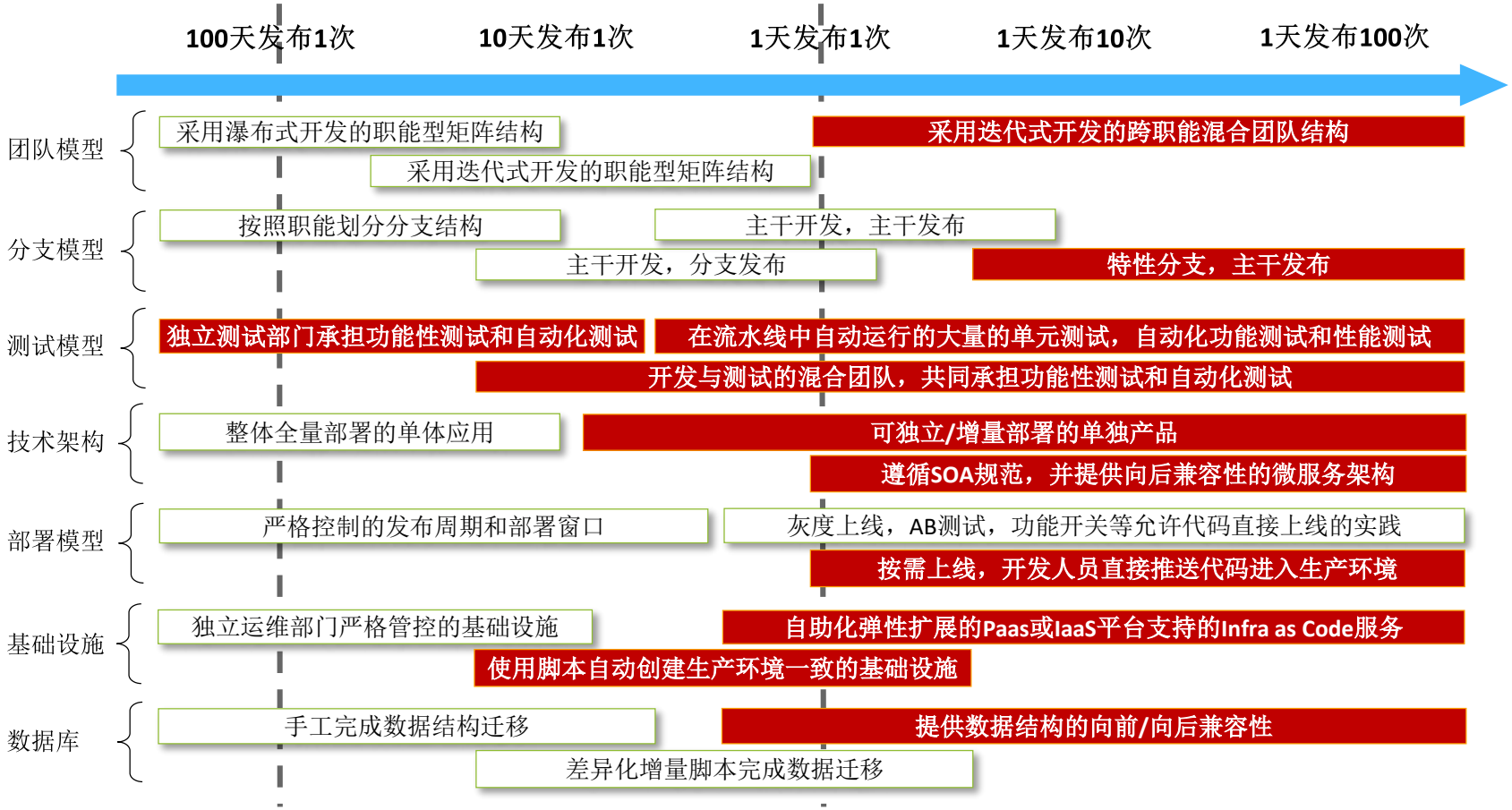
一次构建，多处运行

- 干净的，环境独立的，可迁移的运行平台
- 在多次部署中不必担心依赖，包含环境相关配置
- 隔离性，同时运行不同版本的库和依赖环境，而不用担心他们互相影响
- 自动化测试，集成，打包过程；全部可以通过简单的脚本实现
- 降低与不同应用运行平台的兼容性问题
- 享受VM所提供的隔离性，快照等能力，同时又不被笨重的VM所拖累

配置一次，运行任何应用

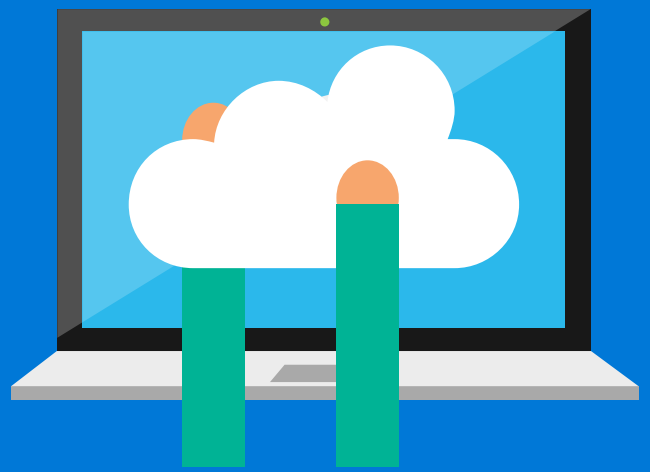
- 让应用生命周期管理变得更加高效，统一可复制
- 提升开发人员的代码质量
- 消除开发，测试，生产和客户定制化环境的差异性
- 为不同职能/技能的人员各司其职提供了条件
- 大大提升CI/CD的可靠性，速度和可复制性
- 应为容器非常轻量，VM所存在的性能，成本，部署和可迁移性问题都迎刃而解

持续交付实施框架 – 容器的定位/价值



LEANSOFT

Agenda



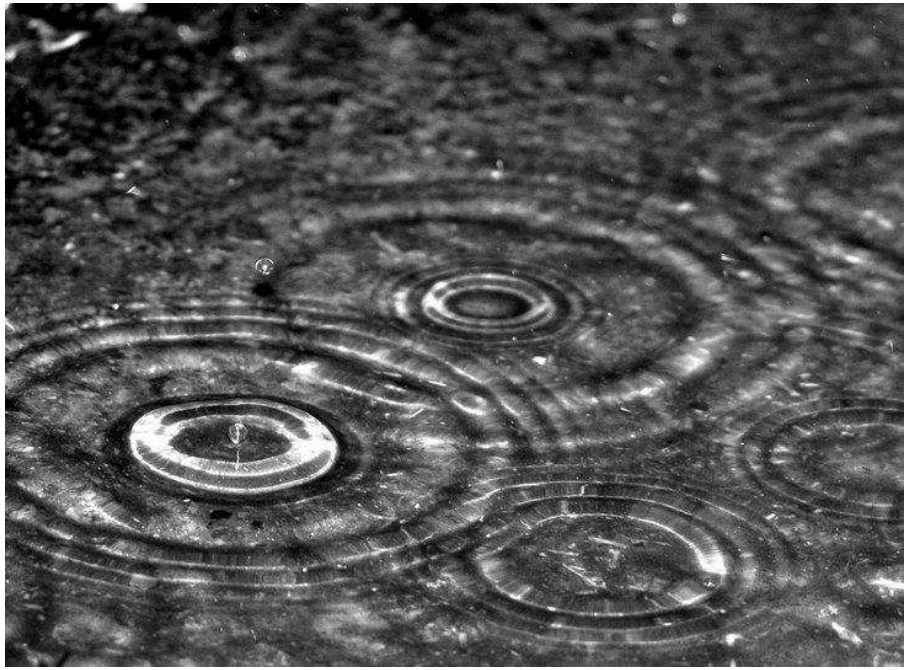
- 100-to-100 持续交付实施框架
- 持续交付就是持续解耦
- 编码工作流
- 测试工作流
- 发布工作流

持续交付的挑战：从ATM取款说起

- 用户场景：
  - 插卡 – 输入密码 – 输入金额 –拿走现金
- 技术实现：
  - ATM机：机械系统控制，智能卡识别，接收用户输入，连接银行系统，监控等等
  - 传输层网络：数据加密，数据完整性，监控
  - 核心银行系统：账户认证，账务信息，审计，风控等等
  - 等等
- 团队：数十个
- 人员：上百个

持续交付的挑战：系统耦合

- 一个系统的修改带来涟漪效应，受影响系统随着系统复杂度增加而呈级数增长
- 被影响系统可能还会造成二次涟漪效应
- 给需求规划，架构设计，开发过程，测试过程...等整个软件开发过程造成极大的困扰



- 如何解决？

持续交付的挑战：软件开发中的三级耦合

代码级耦合

一个开发人员的修改即可影响整个系统

团队规模 < 20

组件级耦合

延迟影响至运行时

多团队协作成为可能

接口定义不通用

无法跨技术栈使用

服务级耦合

延迟影响至生产环境

多团队，多技术栈协作成为可能

接口显视定义

服务自治

团队自由度，业务敏捷能力，交付速度，质量控制，系统复杂度，运维复杂度

容器的价值

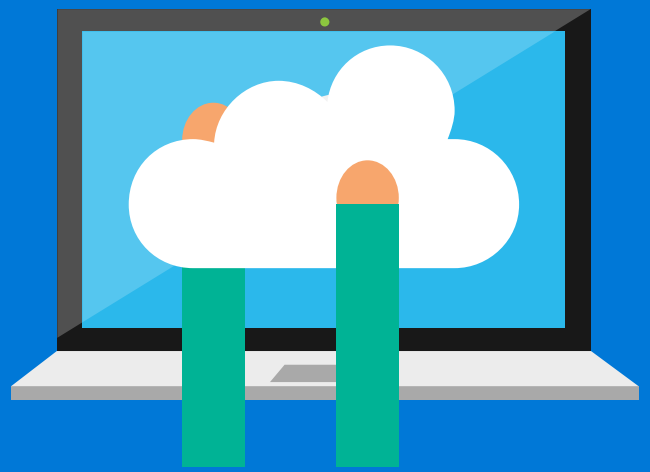
一次构建，多处运行

- 干净的，环境独立的，可迁移的运行平台
- 在多次部署中不必担心依赖，包含环境相关配置
- 隔离性，同时运行不同版本的库和依赖环境，而不用担心他们互相影响
- 自动化测试，集成，打包过程；全部可以通过简单的脚本实现
- 降低与不同应用运行平台的兼容性问题
- 享受VM所提供的隔离性，快照等能力，同时又不被笨重的VM所拖累

配置一次，运行任何应用

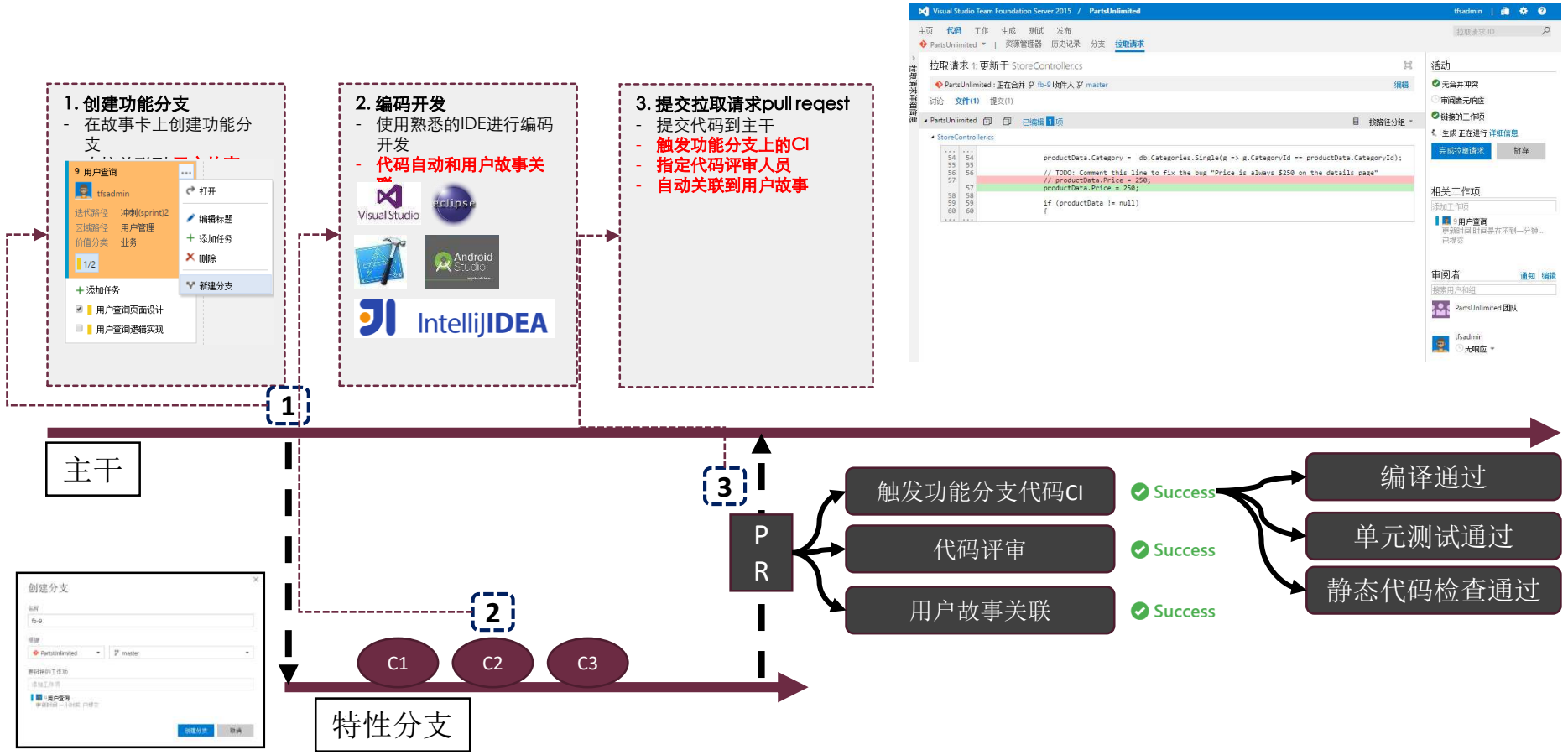
- 让应用生命周期管理变得更加高效，统一可复制
- 提升开发人员的代码质量
- 消除开发，测试，生产和客户定制化环境的差异性
- 为不同职能/技能的人员各司其职提供了条件
- 大大提升CI/CD的可靠性，速度和可复制性
- 应为容器非常轻量，VM所存在的性能，成本，部署和可迁移性问题都迎刃而解

Agenda



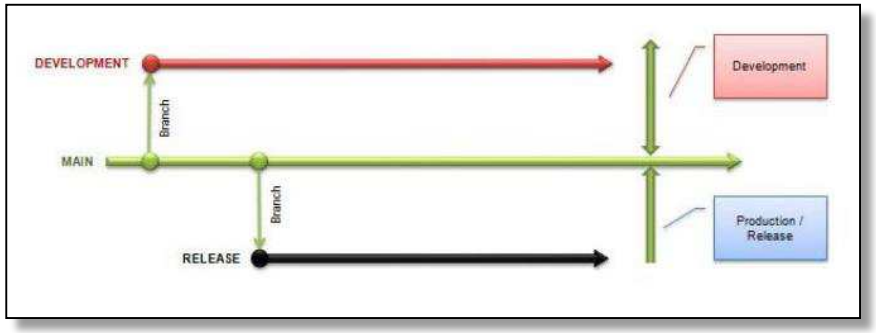
- 100-to-100 持续交付实施框架
- 持续交付就是持续解耦
- 编码工作流
- 测试工作流
- 发布工作流

编码流程 – 特性分支 + PR + 质量门禁

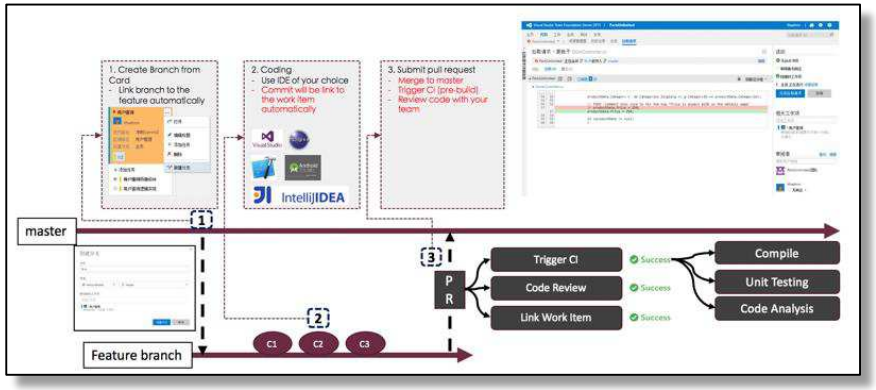


特性分支模型的优势 - 与传统MDR模型的比较

- 传统 M-D-R 模型
  - 面向内部流程
  - 与阶段绑定
  - 无法按照交付挑选代码修改
- 特性模型
  - 面向外部交付
  - 与业务场景绑定
  - 延迟决策



Traditional M-D-R Dodel



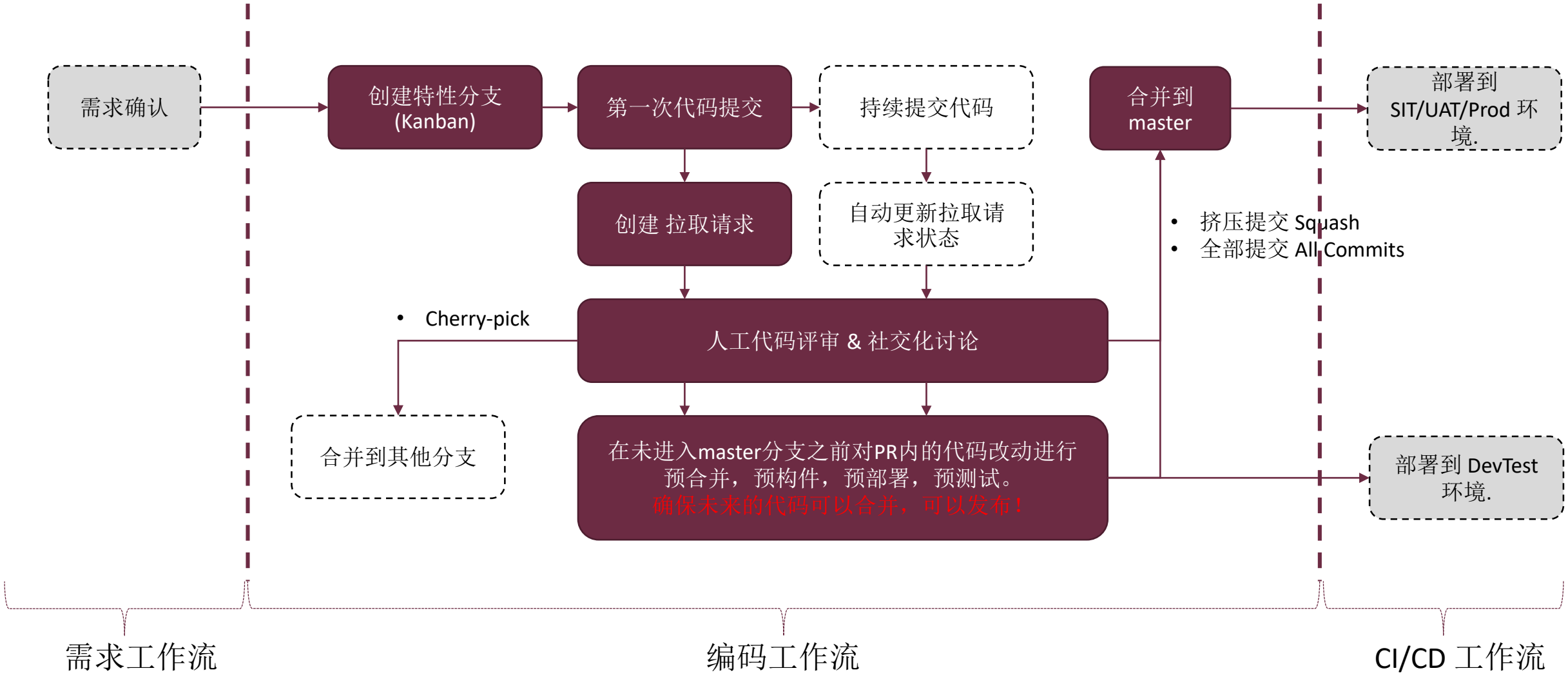
Feature Branch Model

LEANSOFT

特性分支更适应这些场景

- 并行开发，团队解耦
- 按业务场景进行交付
- 独立测试每个特性
- 保持主干代码的整洁
- 简化CI流程
  - 为每个feature建立独立的Ci不在需要在每个feature branch上单独配置
  - 及时与生产代码进行集成

拉取请求 工作流
– 人工代码评审 + CI自动化检查 + CD 自动化部署

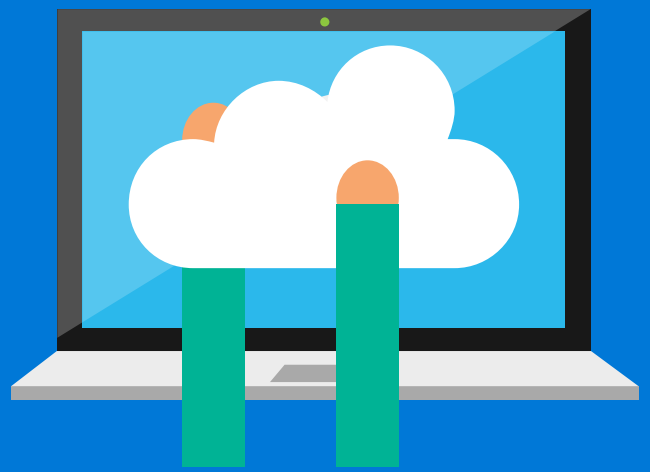


LEANSOFT

为什么要使用 拉取请求 Pull Request?

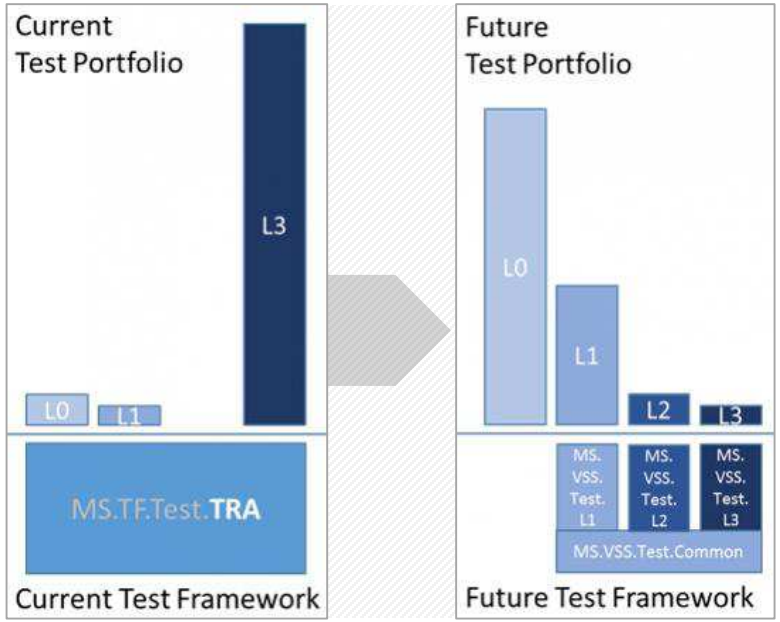
- 审核改动，不要审核提交
- 可视化变更过程
- 预构建（在代码没有合并之前验证合并后的代码）

Agenda



- 100-to-100 持续交付实施框架
- 持续交付就是持续解耦
- 编码工作流
- 测试工作流
- 发布工作流

测试模型：从集成测试向单元测试转变



- 原则

测试应用在最低代码层级编写
编写一次，可在所有环境运行（包括生产环境）
可测试性是设计的重要目标
将测试代码看做生产代码的一部分，仅保留可以稳定运行的测试代码
为测试提供可自助获取的共享资源

L0: 每次签入，只需要运行时文件就可以运行，在CI中执行，必须迅速可靠
L1: 每次签入，但需要依赖环境资源（如：SQL）
L2: 必须针对“特定的”环境运行，逐步清理。
L3: 直接在生产环境运行

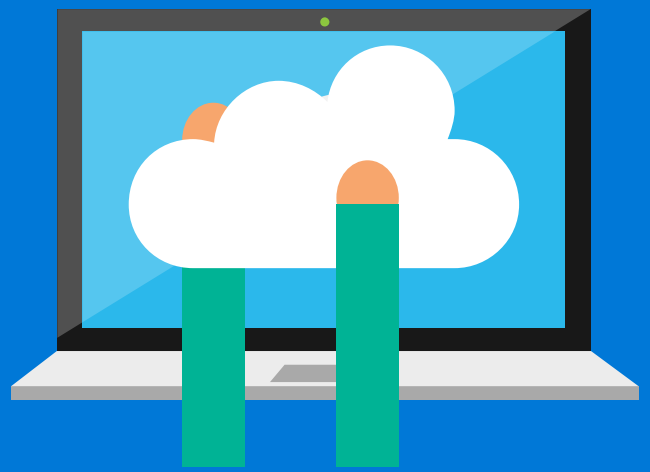
敏捷环境下的测试管理建议（1）

- 测试团队规划
  - 将测试人员与开发人员组成虚拟团队，逐渐转变测试部门与开发部门相互独立的组织架构
  - 统一测试人员与开发人员的KPI导向，使用外向型指标（交付质量），避免或减少内向型指标（测试通过率、bug数量）
- 测试流程规划
  - 建立单元测试，功能测试，UAT测试三级测试机制；并与敏捷迭代节奏相配合进行测试流程组织。
  - 单元测试作为开发人员的任务完成规范一部分，代码签入的同时完成单元测试代码的编写和维护，并通过CI进行检查配合人工代码评审确保此流程的实时。
  - 功能测试作为测试人员的任务完成规范一部分，与迭代周期相互配合。为了保证测试周期与开发周期尽量一直，需要对需求粒度和需求可测试性在规划过程中给你足够的重视，考虑对测试的影响。同时，在迭代时间分配上留出一定的测试执行和bug修复的预留时间，确保迭代交付物是经过测试可上线的产品。

敏捷环境下的测试管理建议（2）

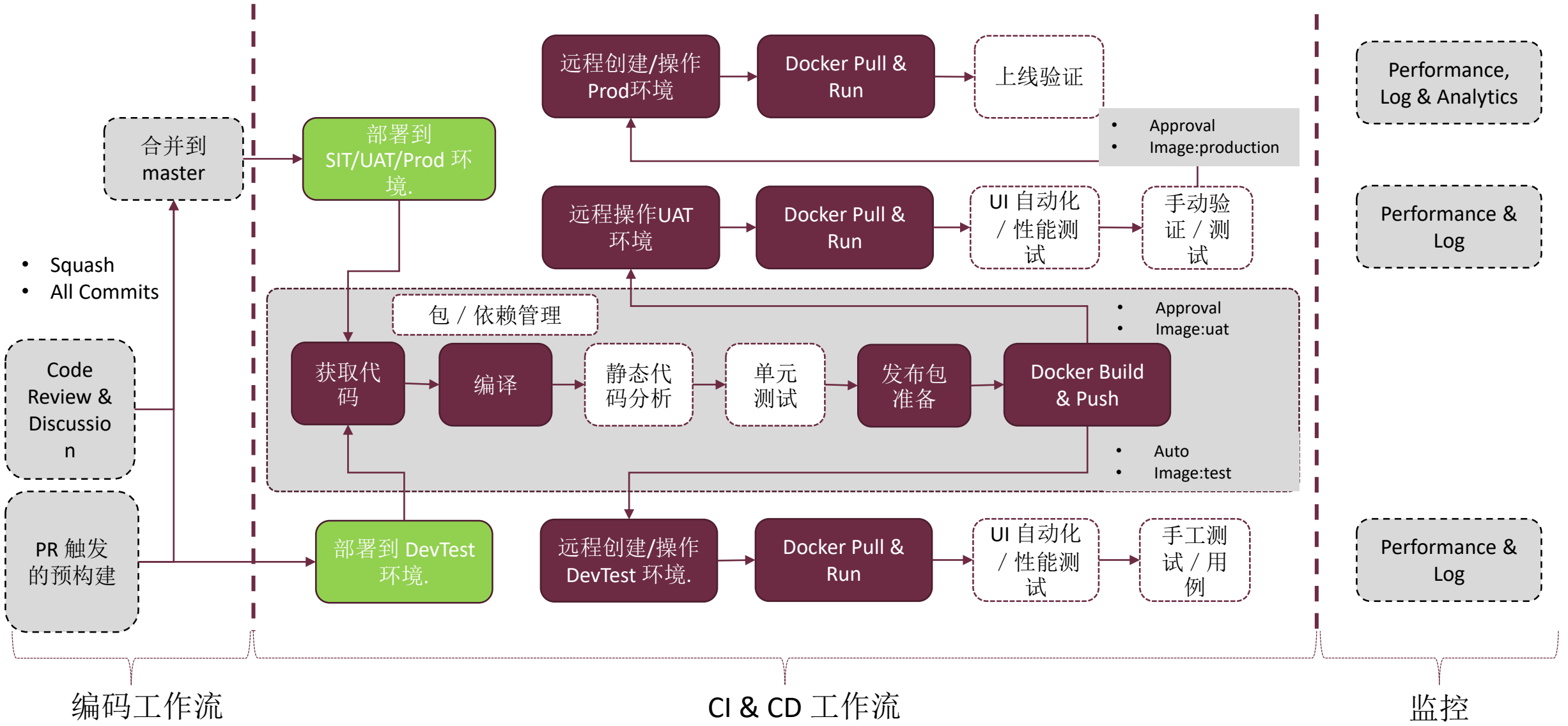
- 测试工程方法
  - 结合持续交付实施框架，对测试工程方法进行持续改进
  - 逐步增加在持续交付流水线（CI/CD）过程中自动化执行测试用例的能力
  - 鼓励开发人员多进行单元测试，减少对手工功能测试的依赖，提升测试自动化能力
  - 通过Infrastructure as Code等DevOps实践改进获取测试环境的效率，逐步做到测试环境可以随用随建，用完销毁；这样做不仅仅可以提升测试运行效率，也会促进开发和运维团队的协作和DevOps实践的推广。
  - 逐步引入AB测试，灰度上线等线上测试机制的，让代码直接流入生产环境，让开发人员脱离“编码完成就是完成任务”的思维模式，建立“代码未进生产就不算做完”的“生产环境优先”意识。
  - 逐步引入容器化技术，为开发和运维创造更优化的协作机制。

Agenda



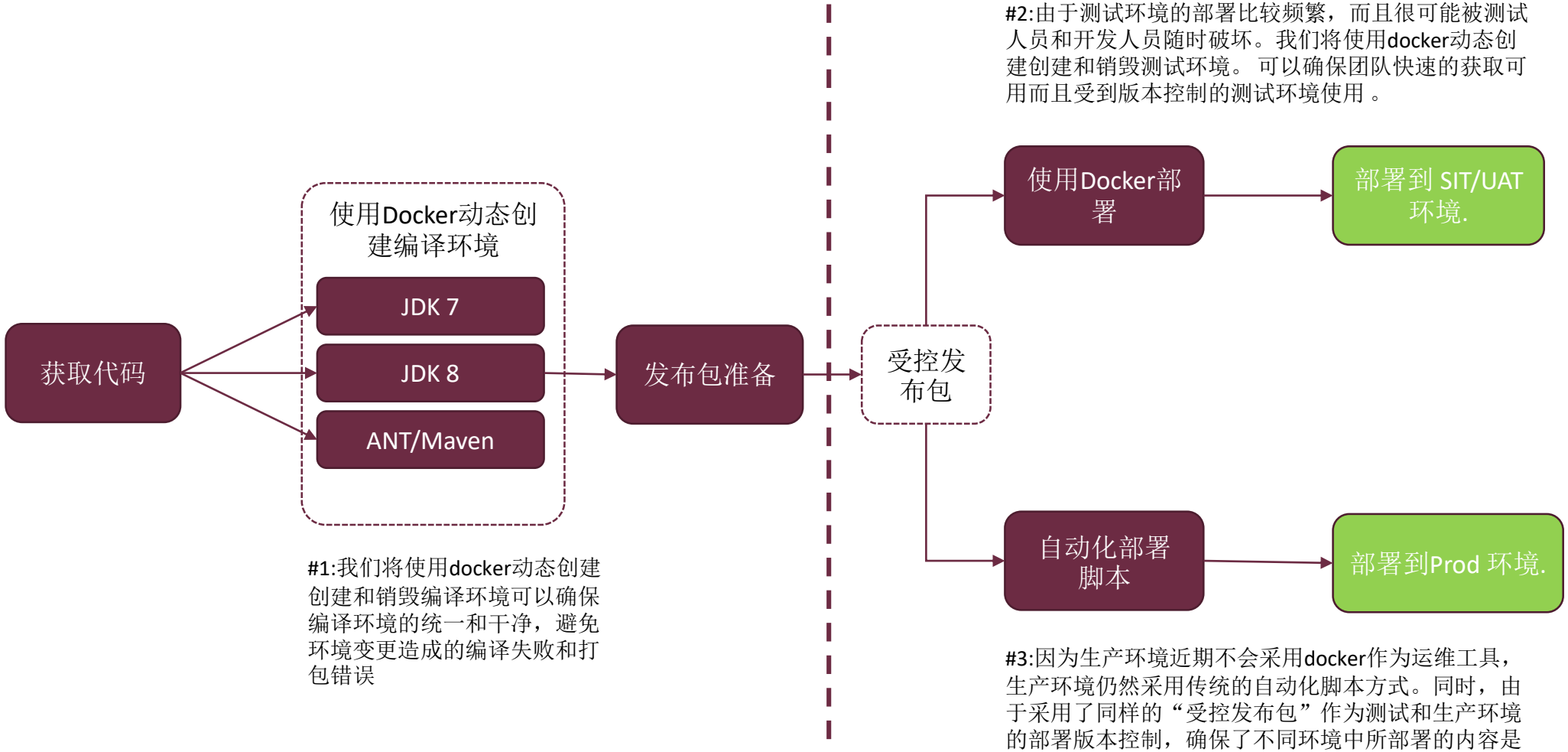
- 100-to-100 持续交付实施框架
- 持续交付就是持续解耦
- 编码工作流
- 测试工作流
- 发布工作流

CI/CD 工作流



LEANSOFT

基于Docker的CI/CD工作流设计思路



#1:我们将使用docker动态创建创建和销毁编译环境可以确保编译环境的统一和干净，避免环境变更造成的编译失败和打包错误

#2:由于测试环境的部署比较频繁，而且很可能被测试人员和开发人员随时破坏。我们将使用docker动态创建创建和销毁测试环境。 可以确保团队快速的获取可用而且受到版本控制的测试环境使用 。

#3:因为生产环境近期不会采用docker作为运维工具，生产环境仍然采用传统的自动化脚本方式。同时，由于采用了同样的“受控发布包”作为测试和生产环境的部署版本控制，确保了不同环境中所部署的内容是一致的。

LEANSOFT 简介

Nobody knows DevOps better than us!

英捷创软科技(北京)有限公司(LEANSOFT)是一家专注于软件工程，敏捷开发和DevOps领域产品开发和服务的解决方案提供商。

公司由15年 **软件研发经验,资深ALM/DevOps专家**创建并任公司首席架构师，至今已经为超过100家不同类型和规模的客户提供过ALM解决方案的咨询和落地服务。

联系我们

137 1126 4760

info@lean-soft.cn

我们致力于为广大开发人员和企业提供最优化的DevOps/敏捷咨询服务和软件工程解决方案。

请扫描右侧二维码加入中国最大的DevOps社区【DevOpsHub】

社区网站 https://devopshub.cn