迭代开发与持续集成

清华大学软件学院 刘强

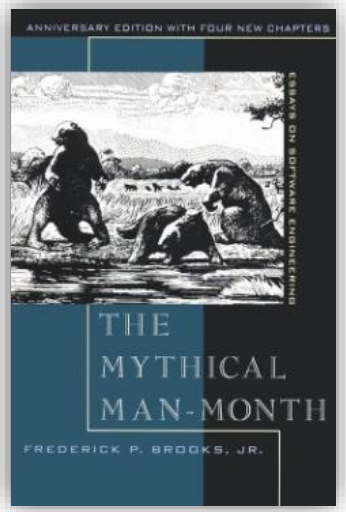
教学提纲

1. 软件配置管理
2. 基于Git 的团队协作开发
3. 持续集成与交付

软件配置管理

There is nothing in this world constant but inconstancy.

—— **Franklin D. Roosevelt**



开发人员交付的是用户满意程度，而不仅仅是实际的产品。用户的实际需要和用户感觉会随着程序的构建、测试和使用而变化。

软件配置管理

- 找不到某个文件的历史版本
- 开发人员使用错误的版本修改程序
- 开发人员未经授权修改代码或文档
- 人员流动，交接工作不彻底
- 无法重新编译某个历史版本
- 因为协同开发或异地开发，版本变更混乱

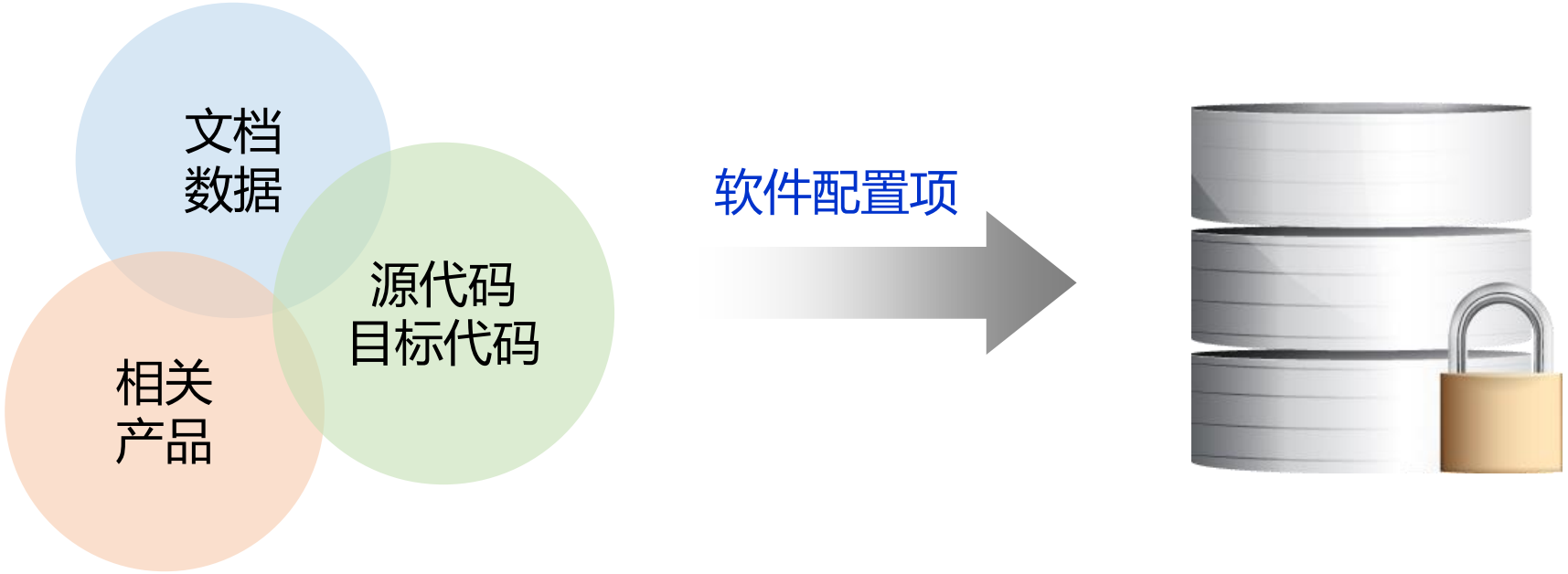
软件配置管理

软件配置管理是一种标识、组织和控制修改的技术，它作用于整个软件生命周期，其目的是使错误达到最小并最有效地提高生产率。

- 记录软件产品的演化过程
- 确保开发人员在软件生命周期的每一个阶段都可以获得精确的产品配置
- 保证软件产品的完整性、一致性和可追溯性

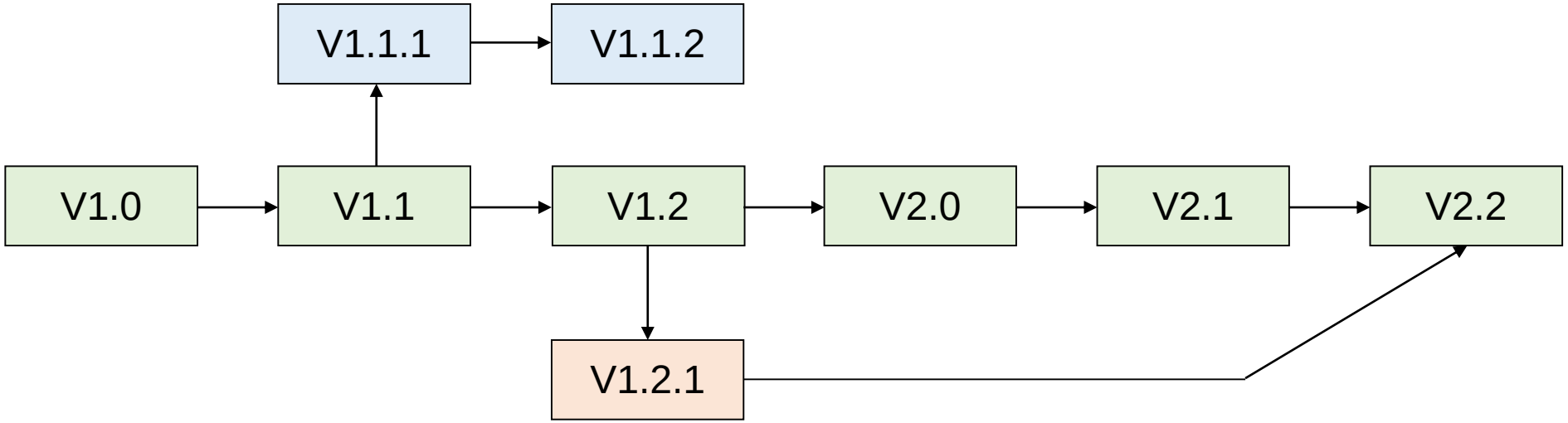
术语：软件配置项

软件配置项（Software Configuration Item，简称SCI）是为了配置管理而作为单独实体处理的一个工作产品或软件。



术语：版本

版本（Version）是在明确定义的时间点上某个配置项状态；随着功能的增加，修改或删除，配置项的版本随之演变。



术语：版本

版本以版本号进行标识，即为了简略表达特定版本的目的和意义，方便区分不同的版本，我们使用版本号这个概念。

版本号格式：X.Y.Z

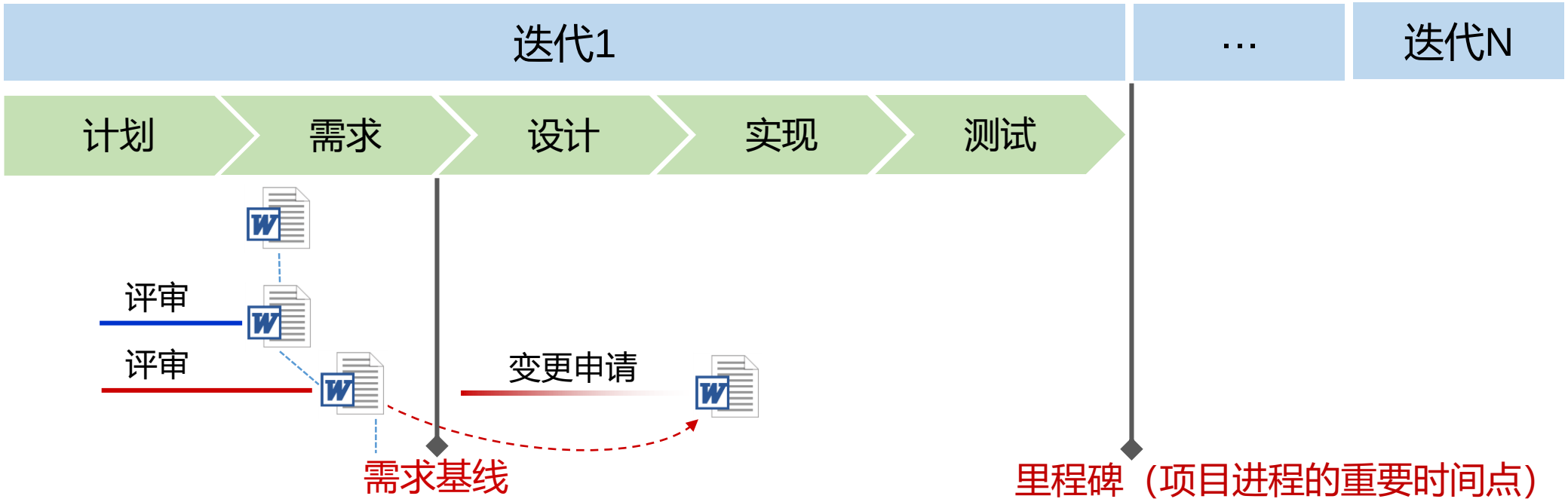
微信 WeChat

Version 7.0.8

- X是主版本：当配置项版本升级幅度比较大，或者有重大架构变更时，允许增大X值。
- Y是次版本：如果配置项的版本升级幅度比较小，有一些功能增加或变化，一般只增大Y值， X值保持不变。
- Z是增量版本：如果只是配置项上的修改，主要是修复一些缺陷，一般只增大Z值，X.Y值保持不变。

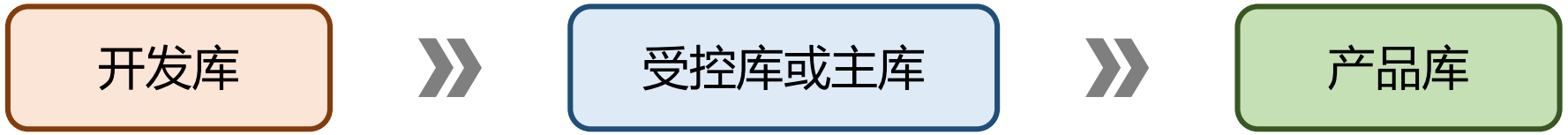
术语：基线

基线（Baseline）是软件配置项的一个稳定版本，它是进一步开发的基础，只有通过正式的变更控制过程才能改变。



术语：配置库

配置库（SCM Repository）存储配置管理信息以及软件配置项的版本信息。



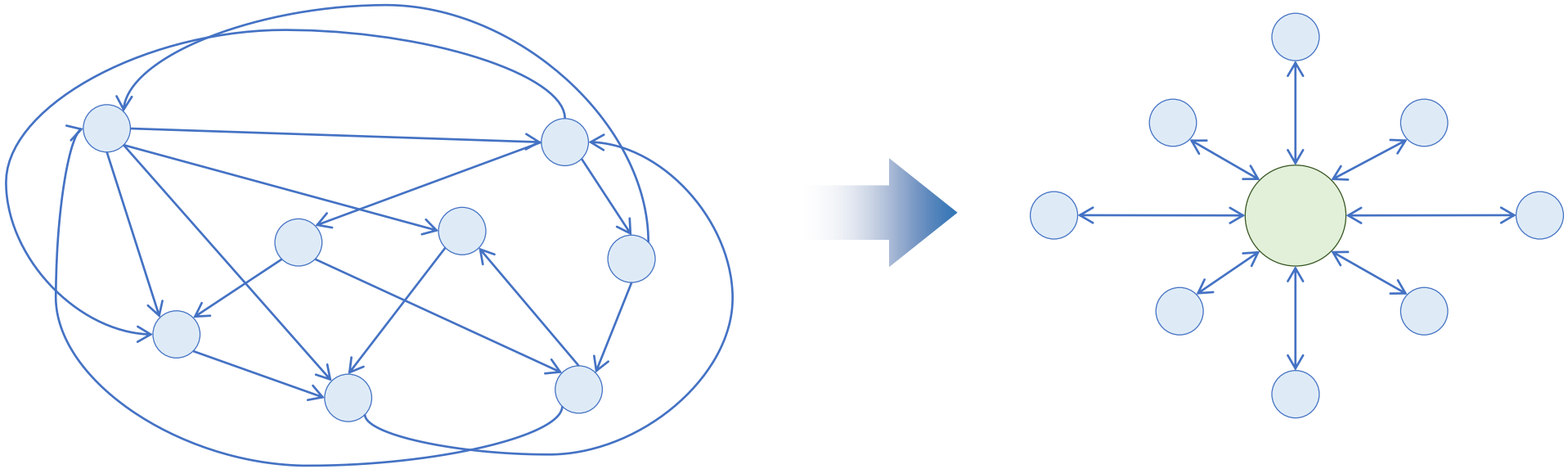
- 存放项目期间处于开发状态的相关文档和代码
- 存放项目组工作期间的相关沟通记录等
- 软件开发人员的工作区，由开发人员控制

- 管理当前基线和控制对基线的变更
- 开发人员在源代码完成并通过自测后将其并入受控库
- 一些开发计划、周报等文档，由专人维护其变更

- 存档各种已分发使用的发布基线
- 在开发产品完成系统测试之后，作为最终产品存入产品库内，等待交付用户或现场安装

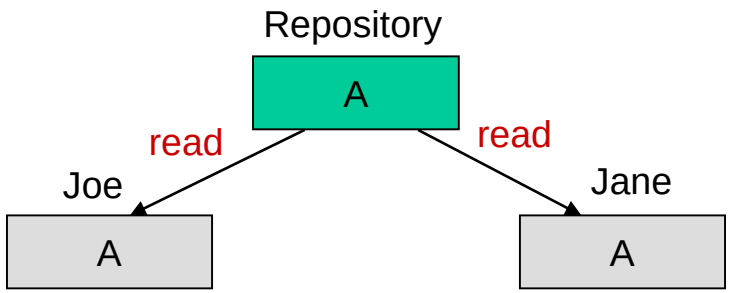
版本控制

场景1：每个程序员各自负责不同的专门模块，没有出现两个程序员修改同一个代码文件的问题。

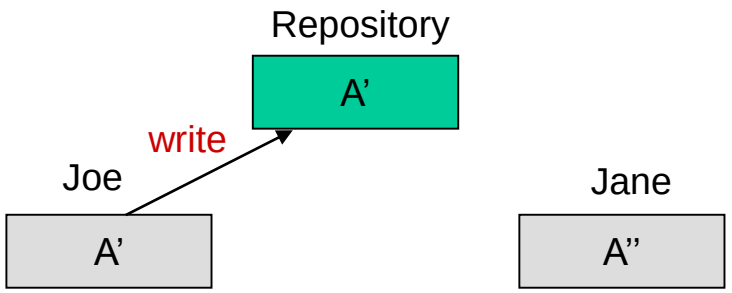


版本控制

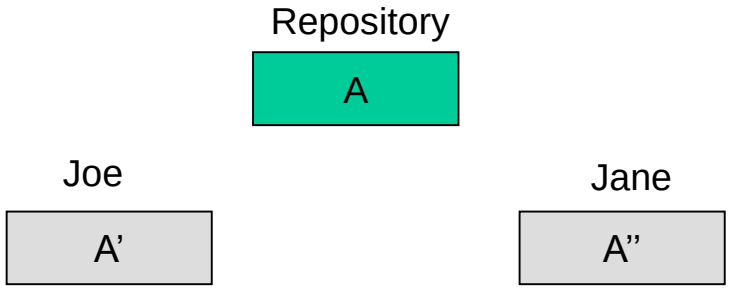
场景2：假设两个程序员同时修改同一个代码文件，就会出现代码覆盖问题。



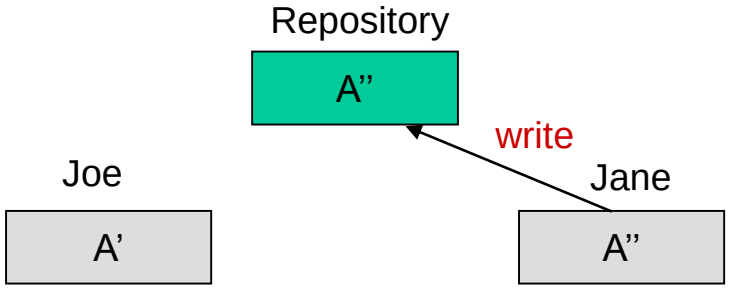
1. 两个程序员读取同一个文件



3. Joe首先发布其版本



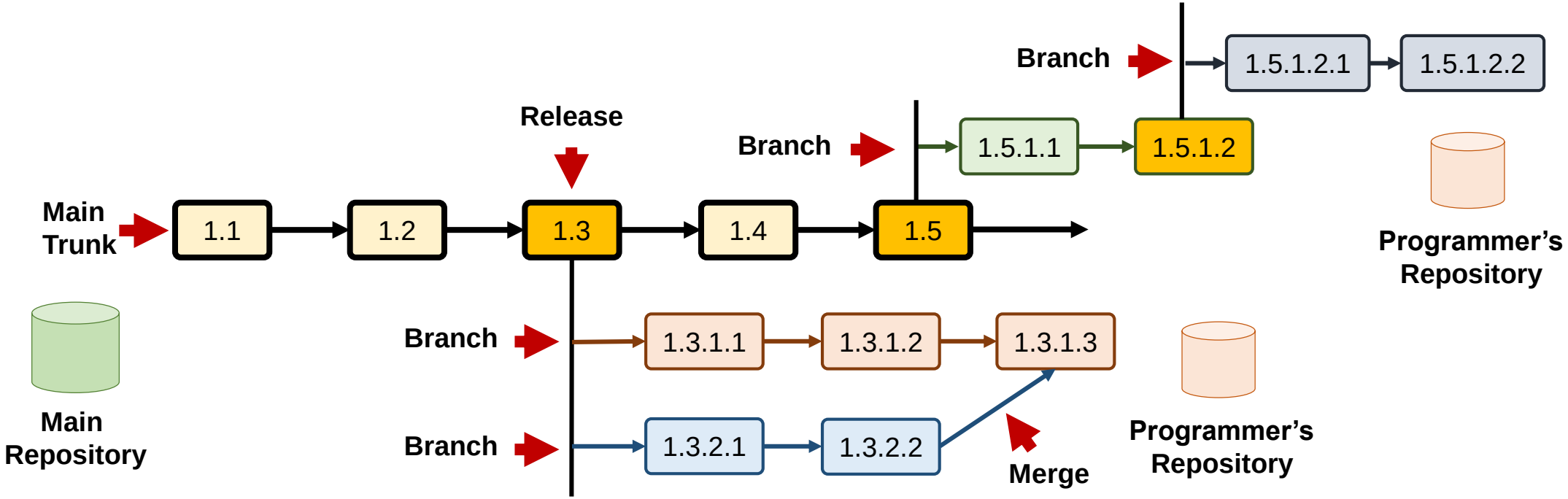
2. 两个程序员开始编辑文件副本



4. Jane意外地覆盖了Joe的版本

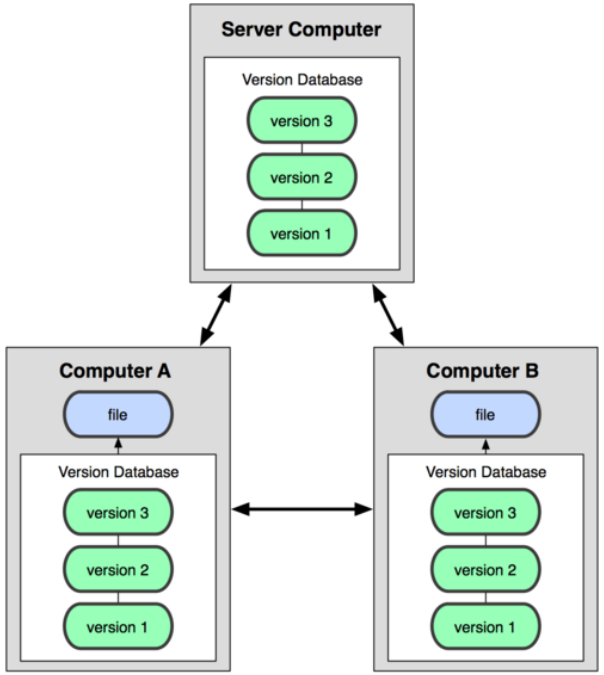
版本控制

版本控制（Version Control）是对系统不同的版本进行标识和跟踪的过程，从而保证软件技术状态的一致性。



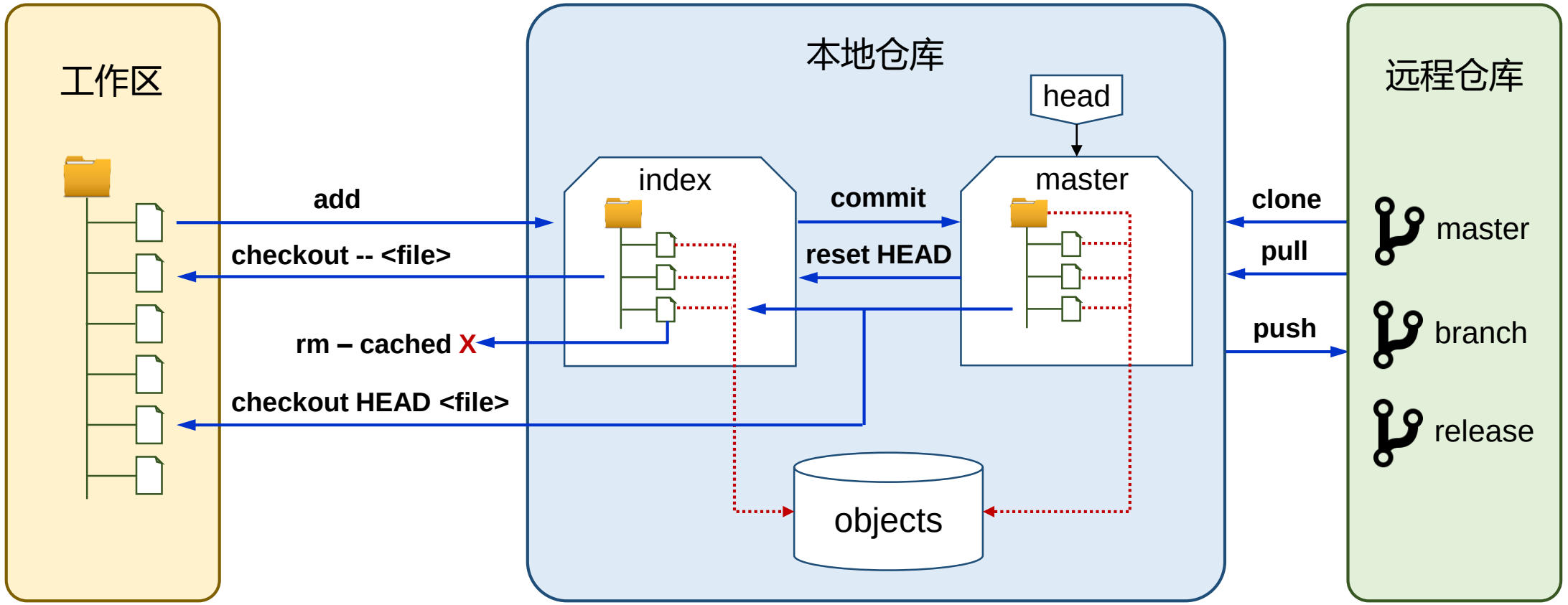
软件配置管理工具 Git

Git 是一个开源的分布式版本控制系统，它最初由 Linux Torvalds 编写，用作 Linux 内核代码的管理，后来在许多其他项目中取得很大的成功。



- 使用中央存储库，但给项目中的每一个开发人员提供完整的项目源代码副本，从而减少对中心仓库的依赖，并支持离线工作。
- 支持快速且可靠地在项目中创建不同的工作集（称为分支），可以跨分支共享更改（称为合并）。
- 可以轻松在开发人员子集之间共享分支和代码更改，这些更改无需先签入中央存储库。

Git 配置库

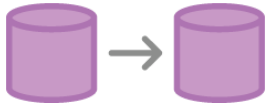

工作区
本地仓库
head
index
master
远程仓库
add
commit
clone
checkout -- <file>
reset HEAD
pull
rm – cached X
push
checkout HEAD <file>
objects
master
branch
release

Git 基本操作

建立一个 Git 仓库

git init

创建一个空的本地仓库



git clone

将远程仓库中的代码克隆到本地

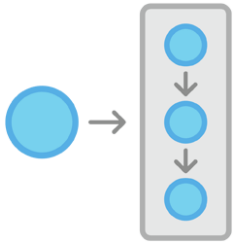
git config

配置或读取相应的工作环境变量

记录快照

git add

把工作区的所有修改提交到暂存区，为提交到本地库准备快照



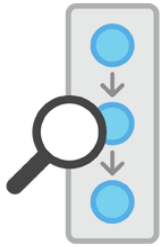
git commit

把暂存的快照提交到本地的版本库中

查看 Git 库

git status

查看工作区和暂存区文件的状态

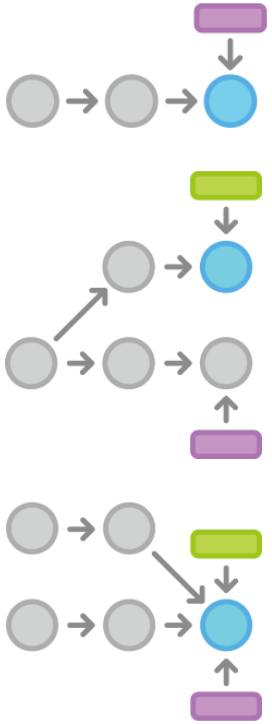


git log

查看提交历史

Git 基本操作

Git 分支



git branch

用于分支操作，包括创建一个新的分支、查看分支等

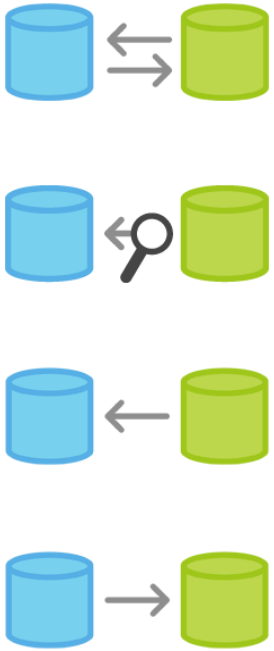
git checkout

将某个文件的历史版本检出到工作区，或者切换到指定分支

git merge

把一些分支内容合并到指定分支

远程仓库



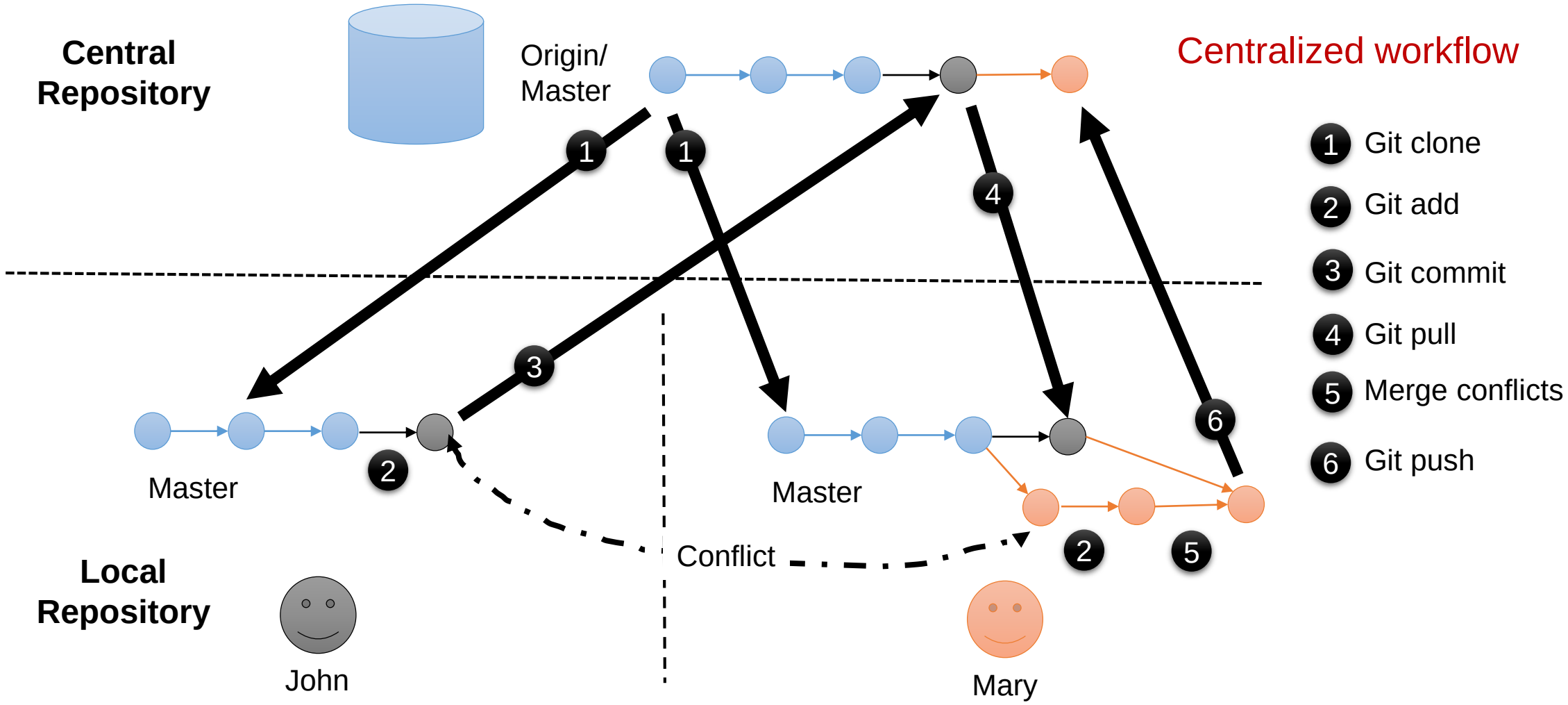
git remote

git fetch

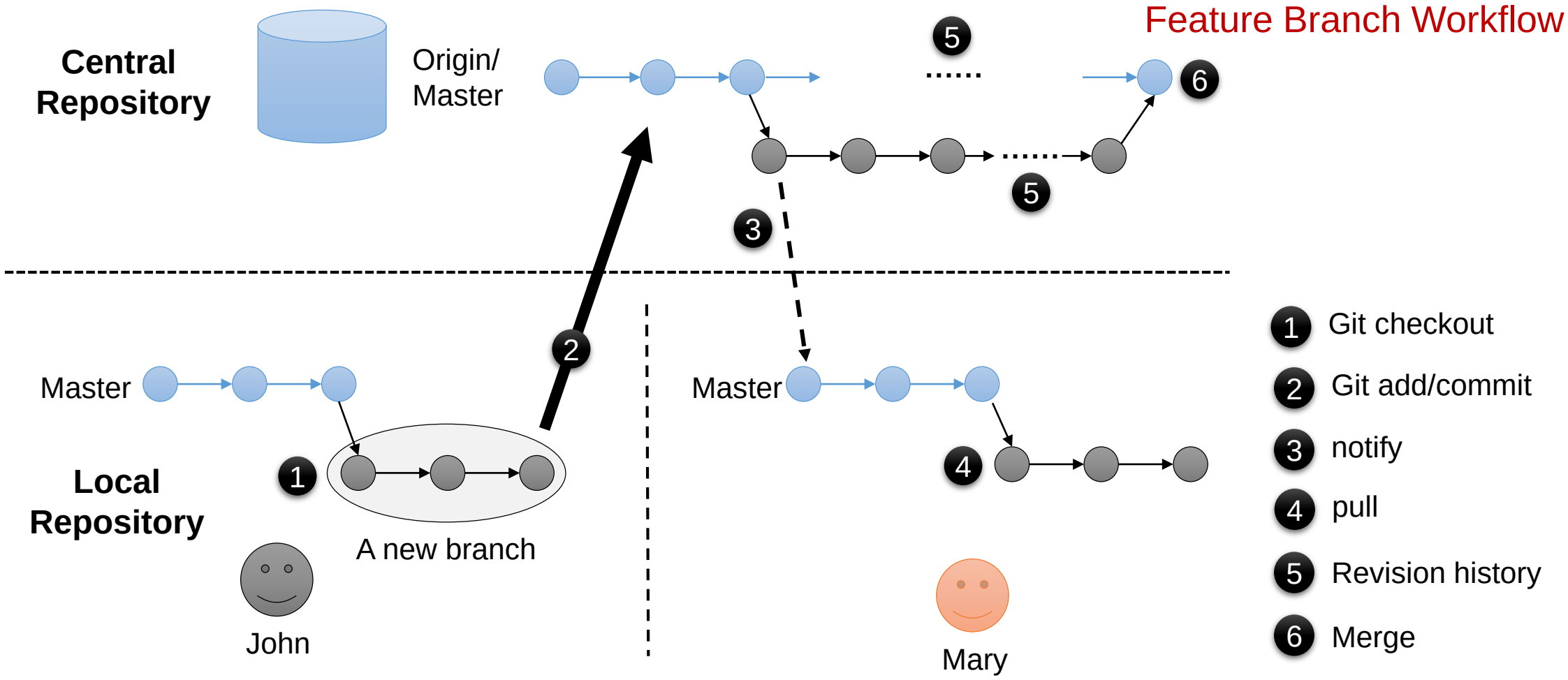
git pull

git push

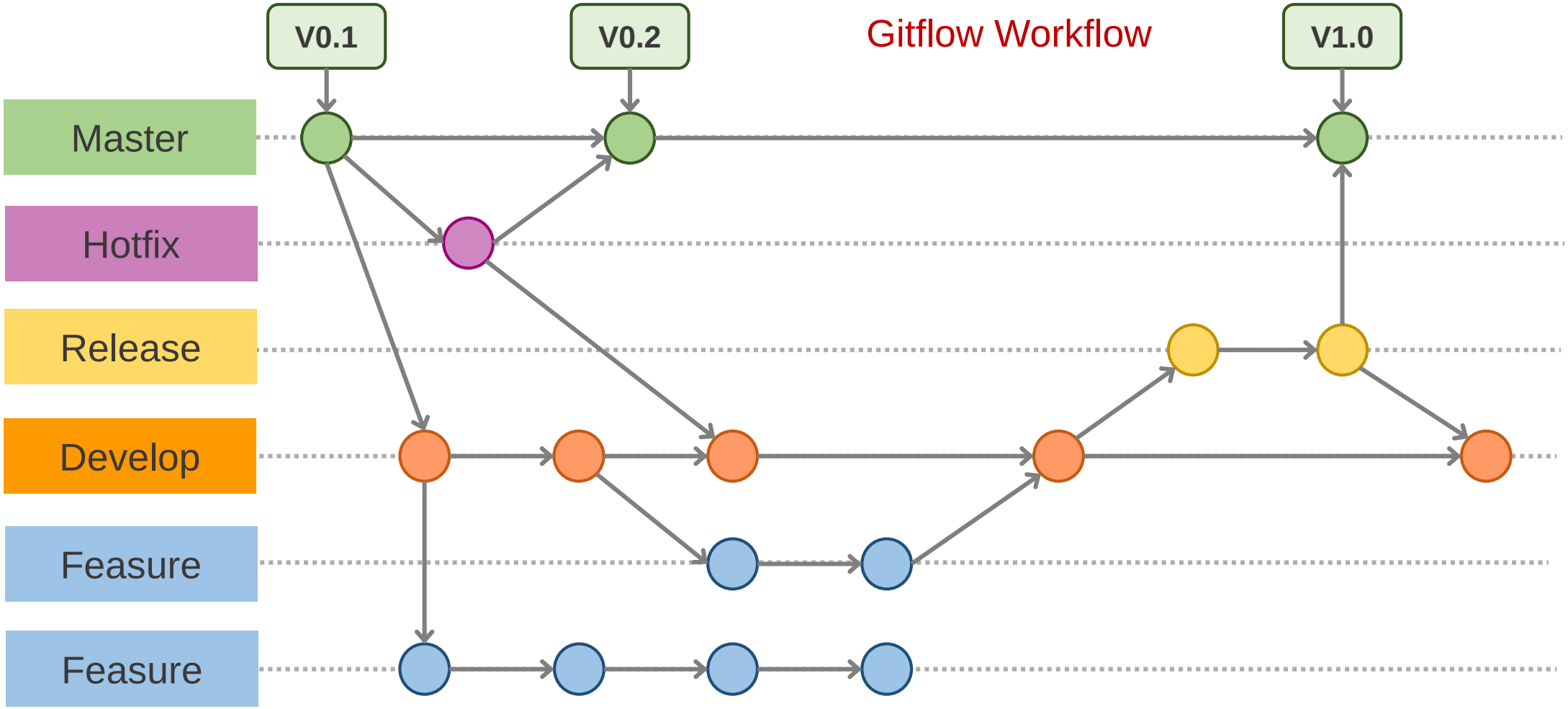
Git 团队协作模式


Central Repository
Origin/ Master
Centralized workflow
1 Git clone
2 Git add
3 Git commit
4 Git pull
5 Merge conflicts
6 Git push
Master
Master
Conflict
Local Repository
John
Mary

Git 团队协作模式


Feature Branch Workflow
Central Repository
Origin/ Master
Master
Local Repository
A new branch
John
Master
Mary
1 Git checkout
2 Git add/commit
3 notify
4 pull
5 Revision history
6 Merge

Git 团队协作模式



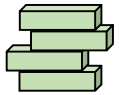
A Successful Git Branching Model

教学提纲

1. 软件配置管理
2. 基于Git 的团队协作开发
3. 持续集成与交付

敏捷开发方法

发布产品



阶段1：愿景

描述：产品目标及其与公司战略的一致性
负责人：产品负责人
频率：至少每年一次

阶段7：迭代回顾

描述：通过改善环境和流程来优化效率
负责人：Scrum团队
频率：每个迭代结束时

阶段2：产品路线图

描述：构成产品愿景的产品特性整体视图
负责人：产品负责人
频率：至少每半年一次

阶段6：迭代评审

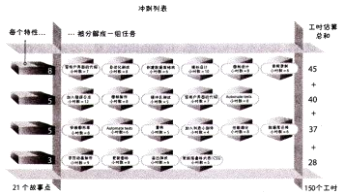
描述：演示可工作产品
负责人：产品负责人和开发团队
频率：每个迭代结束时

阶段3：发布计划

描述：特定产品功能的发布时间计划
负责人：产品负责人
频率：至少每季度一次

阶段5：每日例会

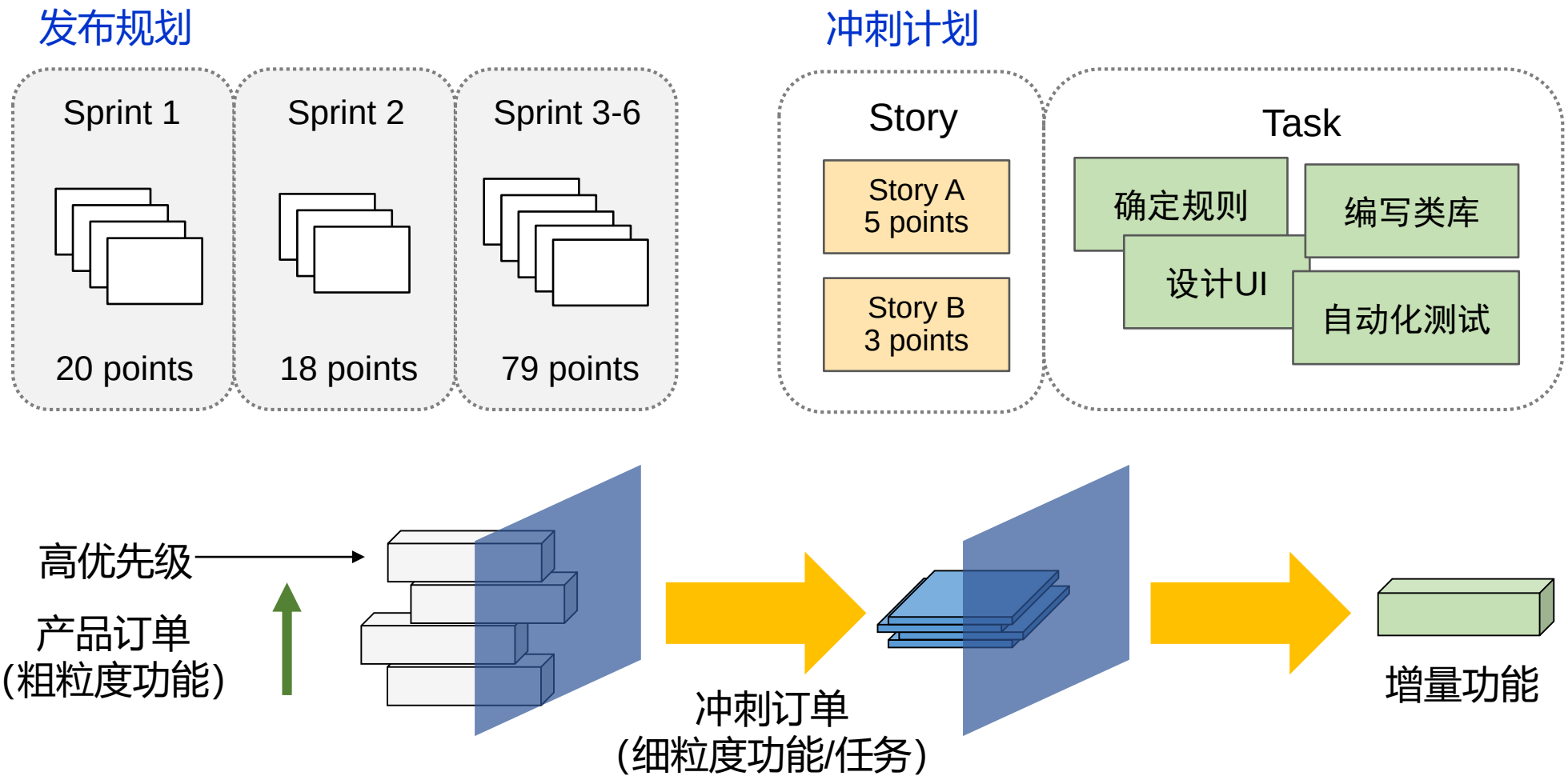
描述：确定并协调当天任务的优先级
负责人：开发团队
频率：每天



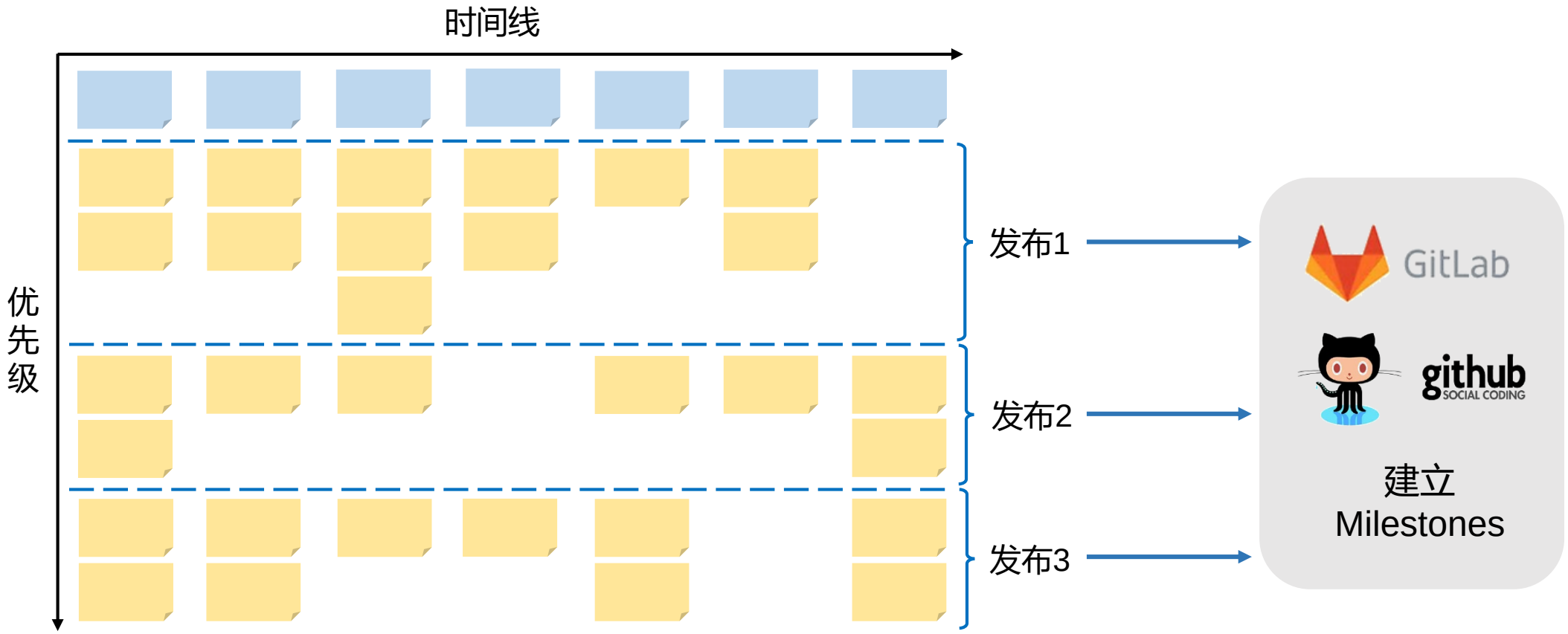
阶段4：迭代计划（冲刺计划）

描述：确定具体的迭代目标和任务
负责人：产品负责人和开发团队
频率：每个迭代开始

敏捷开发方法


发布规划
Sprint 1
Sprint 2
Sprint 3-6
20 points
18 points
79 points
冲刺计划
Story
Story A
5 points
Story B
3 points
Task
确定规则
编写类库
设计UI
自动化测试
高优先级
产品订单
(粗粒度功能)
冲刺订单
(细粒度功能/任务)
增量功能

基于 Git 的迭代开发过程



使用 Issue 管理软件项目

Issue 是一项待完成的工作，通常与系统的改进相关。

- 一个软件的 bug
- 一项功能建议
- 一项待完成的任务
- 文档缺失的报告

每个 Issue 应该包含该问题的所有信息和历史，使得后来的人只看这个 Issue，就能了解问题的所有方面和过程。

使用 Issue 管理软件项目

项目管理

- 指定 Issue 的优先级
- 指定 Issue 所在的阶段
- 分配负责 Issue 的处理人员
- 制定日程
- 监控进度，提供统计

团队合作

- 讨论
- 邮件通知

代码管理

- 将 Issue 关联源码
- 将 Issue 关联代码提交与合并

使用 Issue 管理软件项目

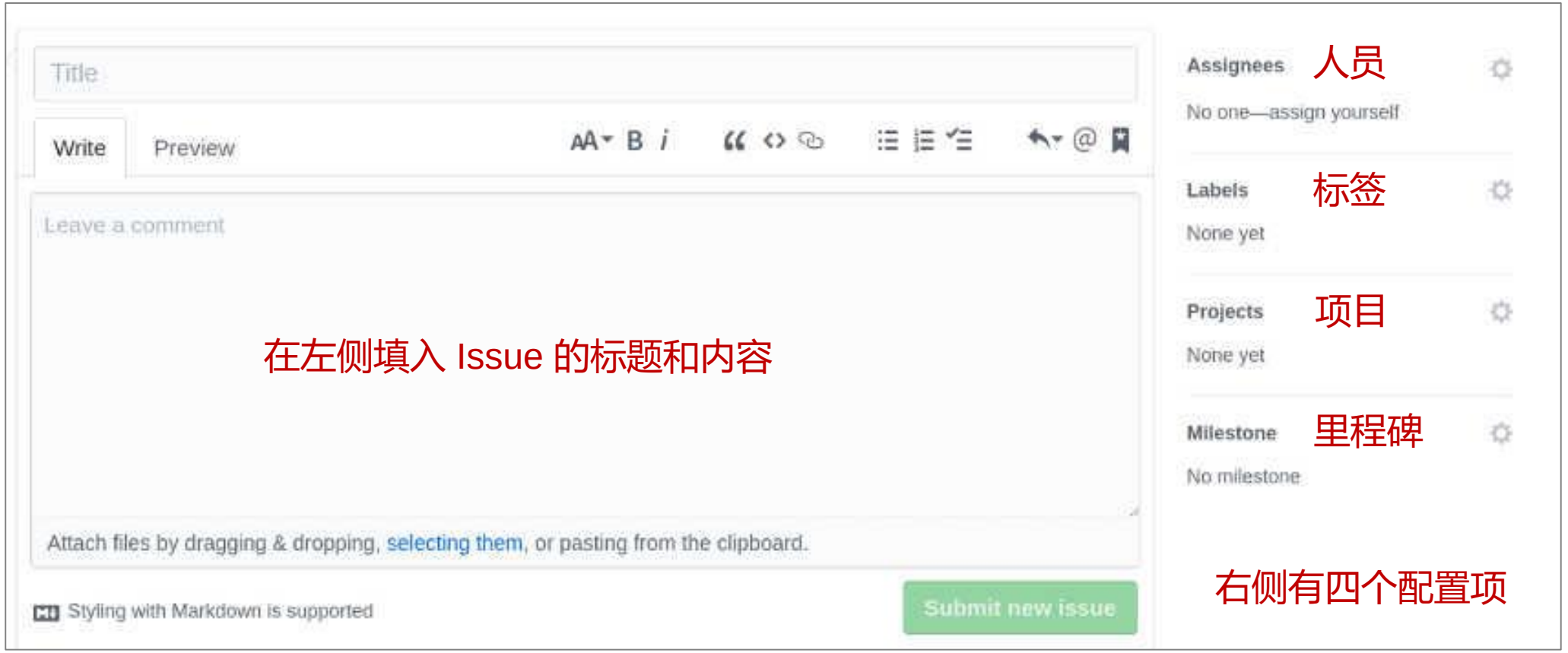
下面以 Github Issues 为例介绍如何使用 Issue

http://www.ruanyifeng.com/blog/2017/08/issue.html

每个 Github 代码仓库都有一个 Issues 面板

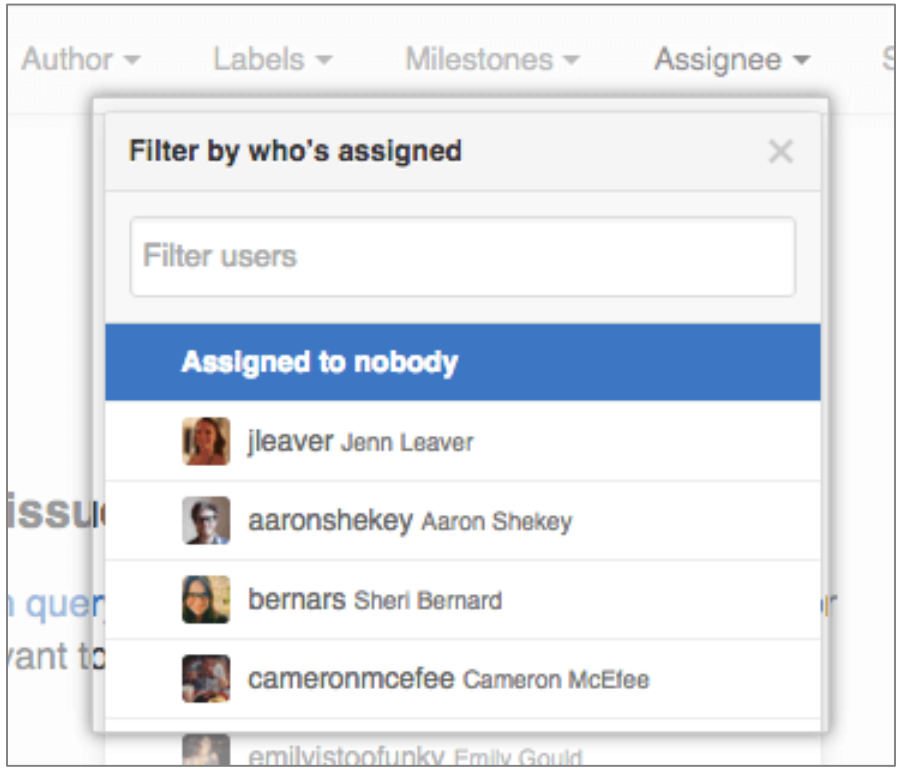
使用 Issue 管理软件项目

在 Issue 面板中，点击“New Issue”按钮，可以新建 Issue。



使用 Issue 管理软件项目

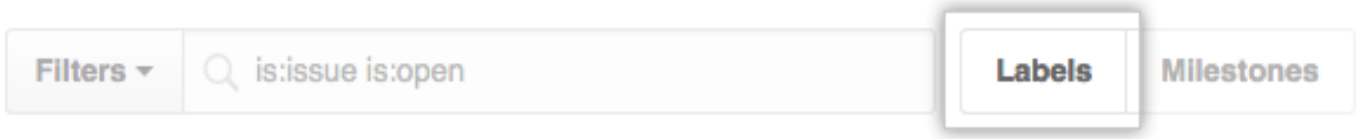
Assignee 选择框用于从当前仓库的所有成员之中，指派某个 Issue 的处理人员。

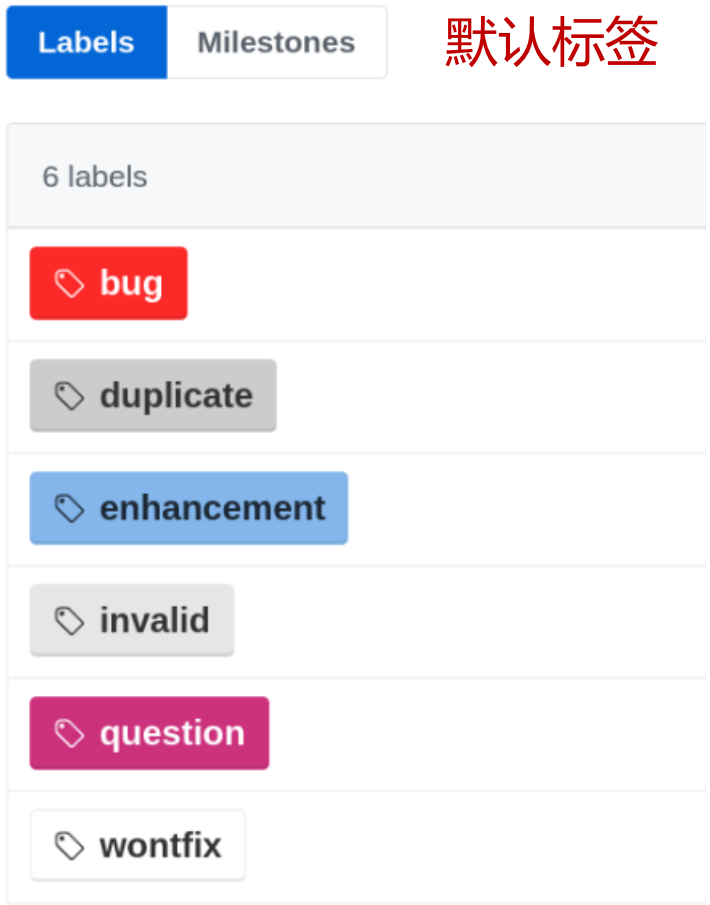


说明：

- 需要先把工作任务分解成Issue，然后把对应任务分配给具体成员
- 课程实验项目检查时将通过Issue核实每个学生的工作情况

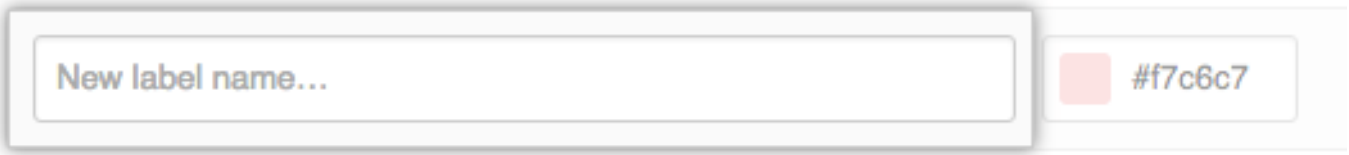
给 Issue 贴上标签，可以有利于分类管理和过滤查看。在 Issues 面板首页，点击 Labels 按钮可以新建标签。





默认标签

如果想新建更多的标签，可以点击 New label 按钮；然后填写标签名，选择标签颜色。



对于大型项目， 每个 Issue 至少应该有两个 Label ，一个表示性质，另一个表示优先级。

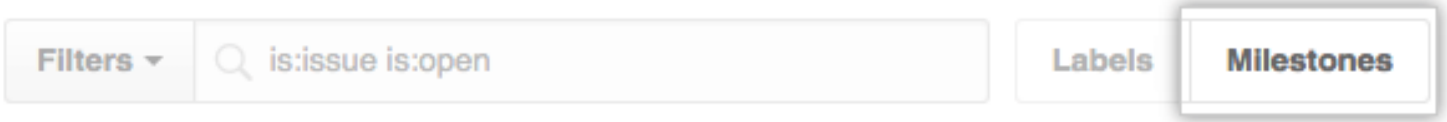
标签性质示例

优先级划分：

- 高优先级（High）：对系统有重大影响，只有解决它之后，才能去完成其他任务。
- 普通优先级（Medium）：对系统的某个部分有影响，用户的一部分操作会达不到预期效果。
- 低优先级（Low）：对系统的某个部分有影响，用户几乎感知不到。
- 微不足道（Trivial）：对系统的功能没有影响，通常是视觉效果不理想，比如字体和颜色不满意。

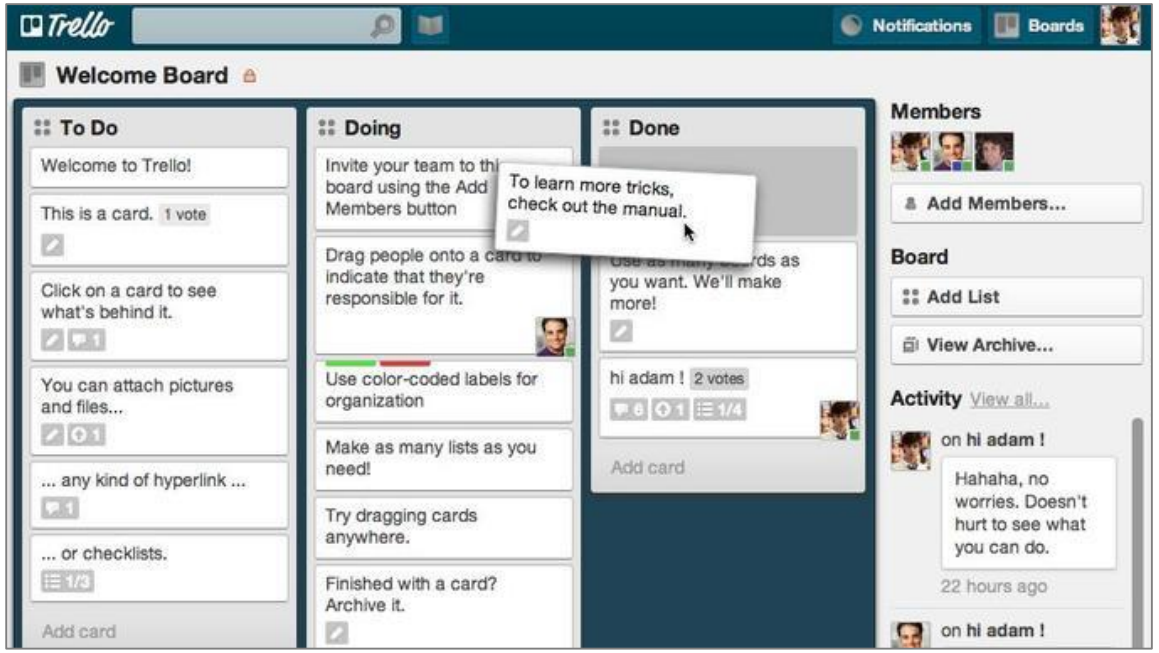
Milestone 称为“里程碑”，用作 Issue 的容器，相关 Issue 可放在一个 Milestone 里。常见的例子是不同的版本（version）和迭代（sprint），都可以做成 Milestone。

在 Issues 面板的首页，点击 Milestones 按钮，新建 Milestone。再接着点击 New milestone 按钮，然后填写 Milestone 的名称和内容，可以指定到期时间。



使用 Issue 管理软件项目

看板是敏捷开发的重要手段，主要用于项目的进度管理。所有需要完成的任务都做成卡片，贴在一块白板上面，这就是看板。



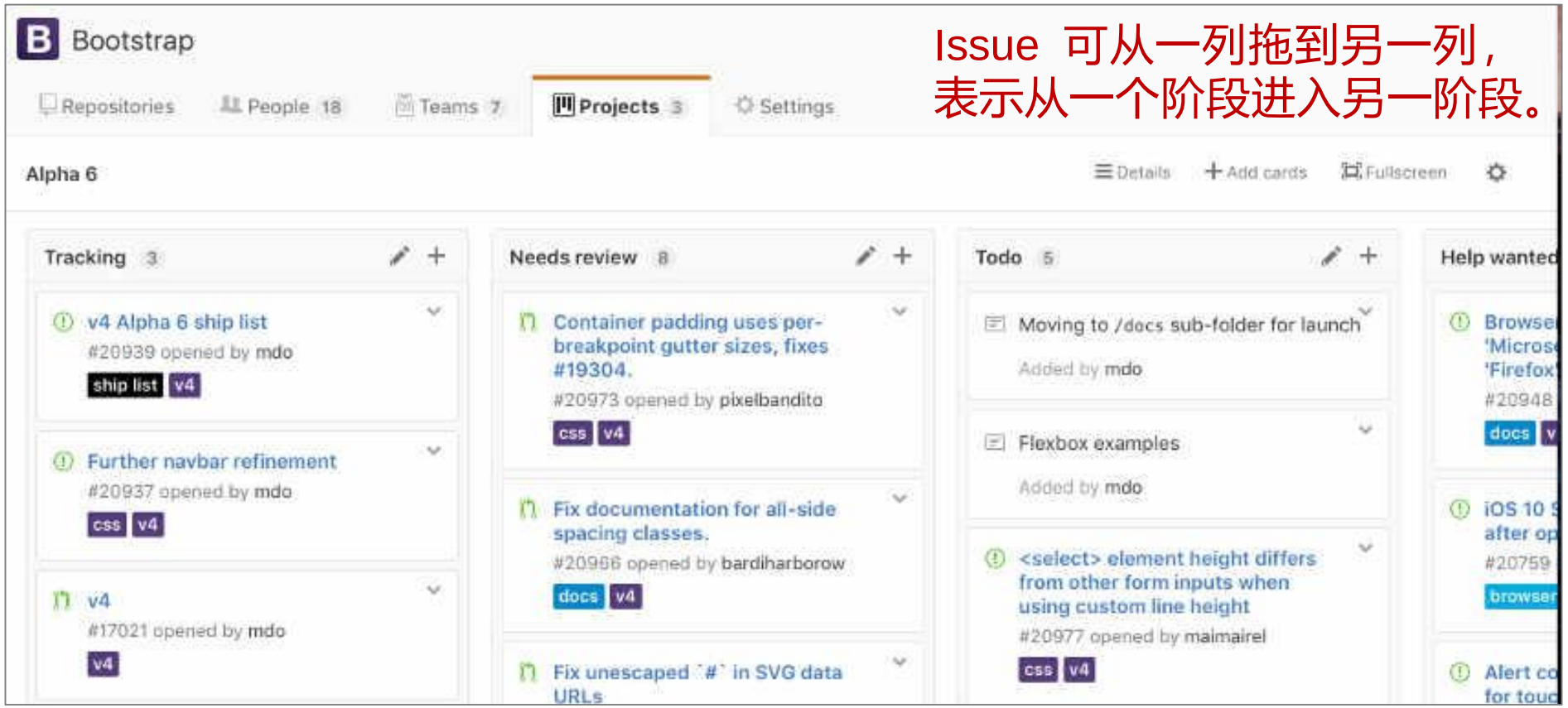
使用 Issue 管理软件项目

按照不同阶段或实际情况，可以将看板分成若干列，最简单的是三列（Todo, Doing, Done）。

Todo	Doing	Done
待开发	开发中	已完成

Todo	Plan	Develop	Test	Deploy	Done
待安排	计划	开发	测试	部署	已完成

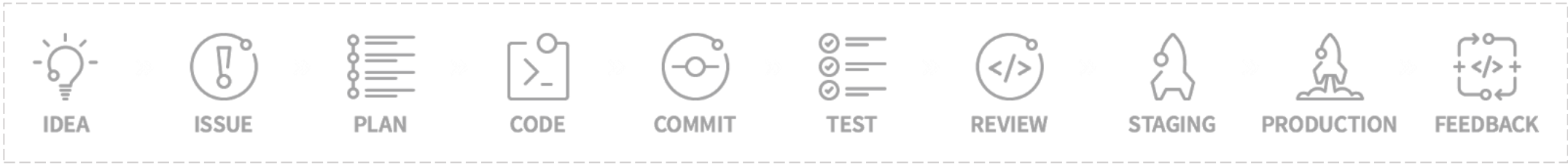
Github 提供 Issue 看板，即在仓库首页进入 Projects 面板，点击 New Project 按钮，新建一个 Project，如“XXV1.0”；再点击 Add column 按钮，为该项目新建若干列。最后，将 Issue 分配到对应的列，则新建成功一个看板视图。



基于Gitlab的敏捷项目开发

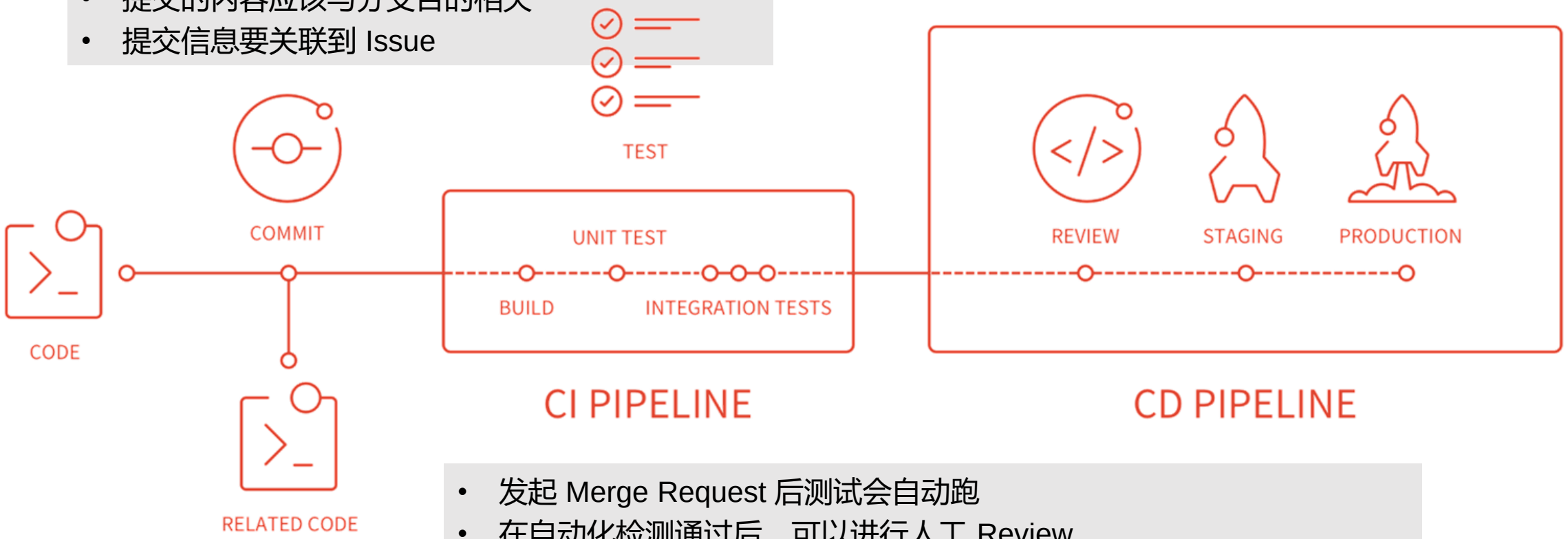
GitLab是一个集代码管理、测试、代码部署于一体的开源应用程序，主要包括有很好的权限控制的Git版本管理系统、代码评审系统、问题跟踪系统、活动反馈系统、Wiki、持续集成CI系统等。

基于GitLab的开发流程



基于Gitlab的敏捷项目开发

- 一个分支关联一个 Issue，一次提交只做一件事情
- 提交的内容应该与分支目的相关
- 提交信息要关联到 Issue

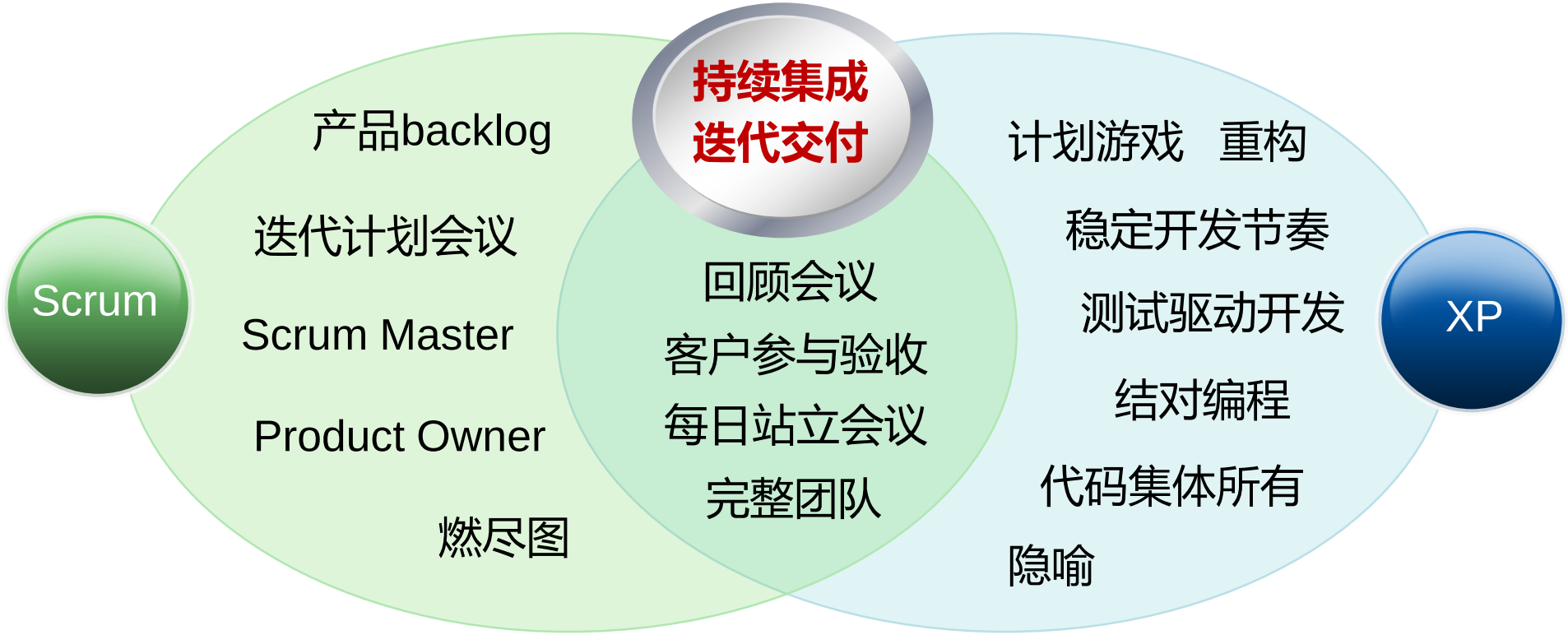


- 发起 Merge Request 后测试会自动跑
- 在自动化检测通过后，可以进行人工 Review
- 人工 Review 通过后可以批准修改，之后由 MR 发起人或组长完成合并

教学提纲

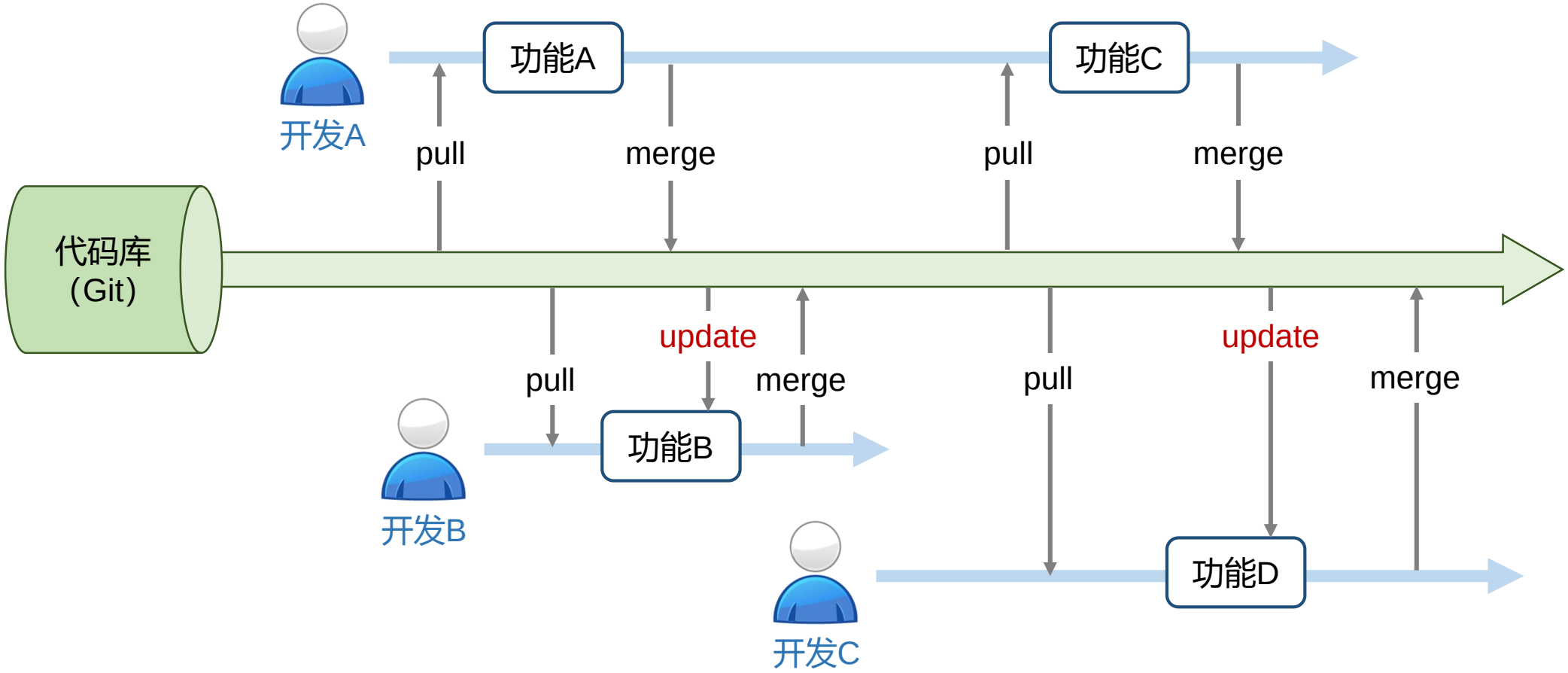
1. 软件配置管理
2. 基于Git 的团队协作开发
3. 持续集成与交付

回顾：敏捷开发方法



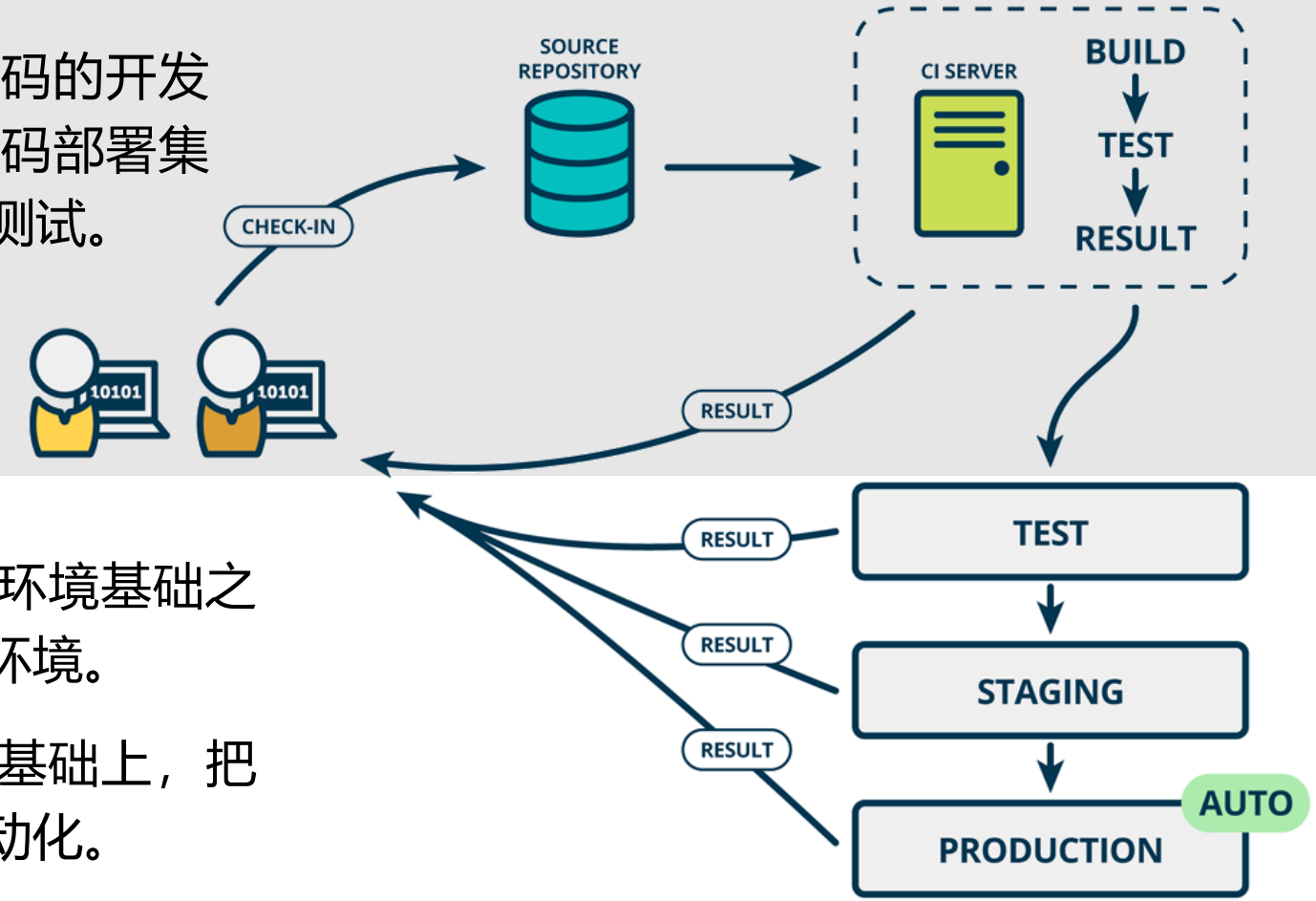
Scrum偏重项目管理　　XP偏重编程实践

持续集成的工作模式


开发A
功能A
功能C
pull
merge
pull
merge
代码库
(Git)
update
update
pull
merge
pull
merge
开发B
功能B
开发C
功能D

持续集成与交付

持续集成是指开发者在代码的开发过程中，可以频繁地将代码部署集成到主干，并进行自动化测试。



持续交付是在持续集成的环境基础之上，将代码部署到预生产环境。

持续部署是在持续交付的基础上，把部署到生产环境的过程自动化。

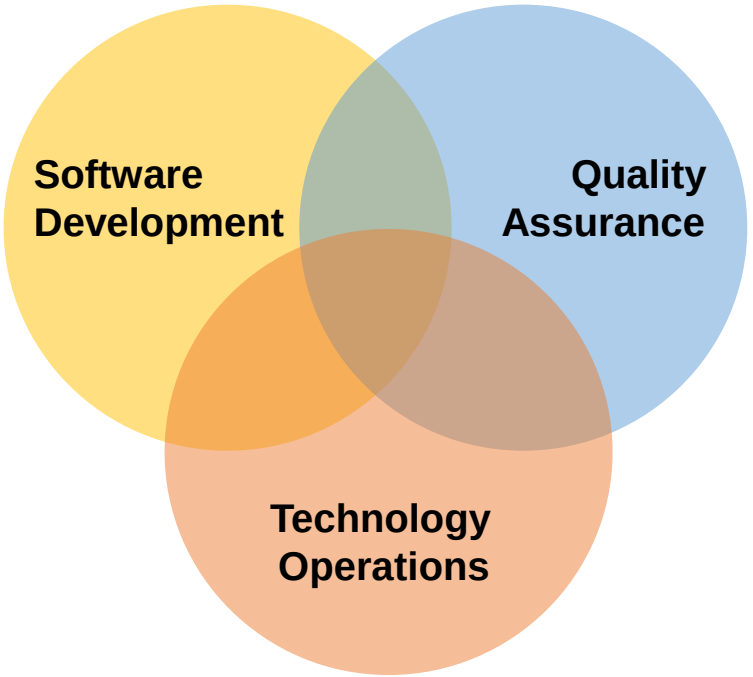
持续集成与交付

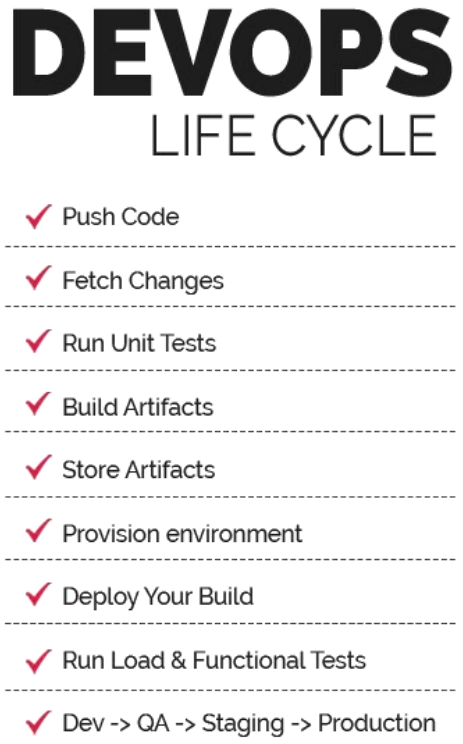
- 一周内平均部署几十次，几乎每个开发人员的每次修改就会导致一次部署。
- 这不仅仅意味着可以更快从用户那里得到使用反馈，更可以迅速对产品进行改进， 更好地适应用户的需求和市场的变化。

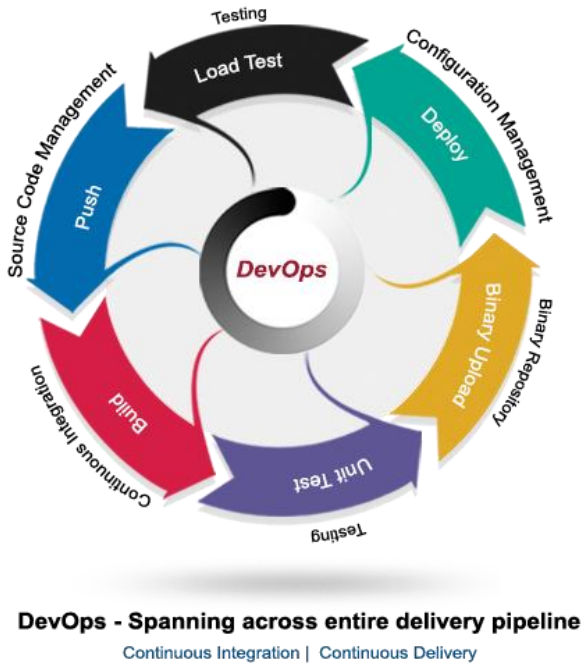
- Facebook有数千名开发和运维人员，成千上万台服务器。
- 平均来说一位运维人员负责500台服务器，他们每天部署两次。

DevOps（Development+Operation）

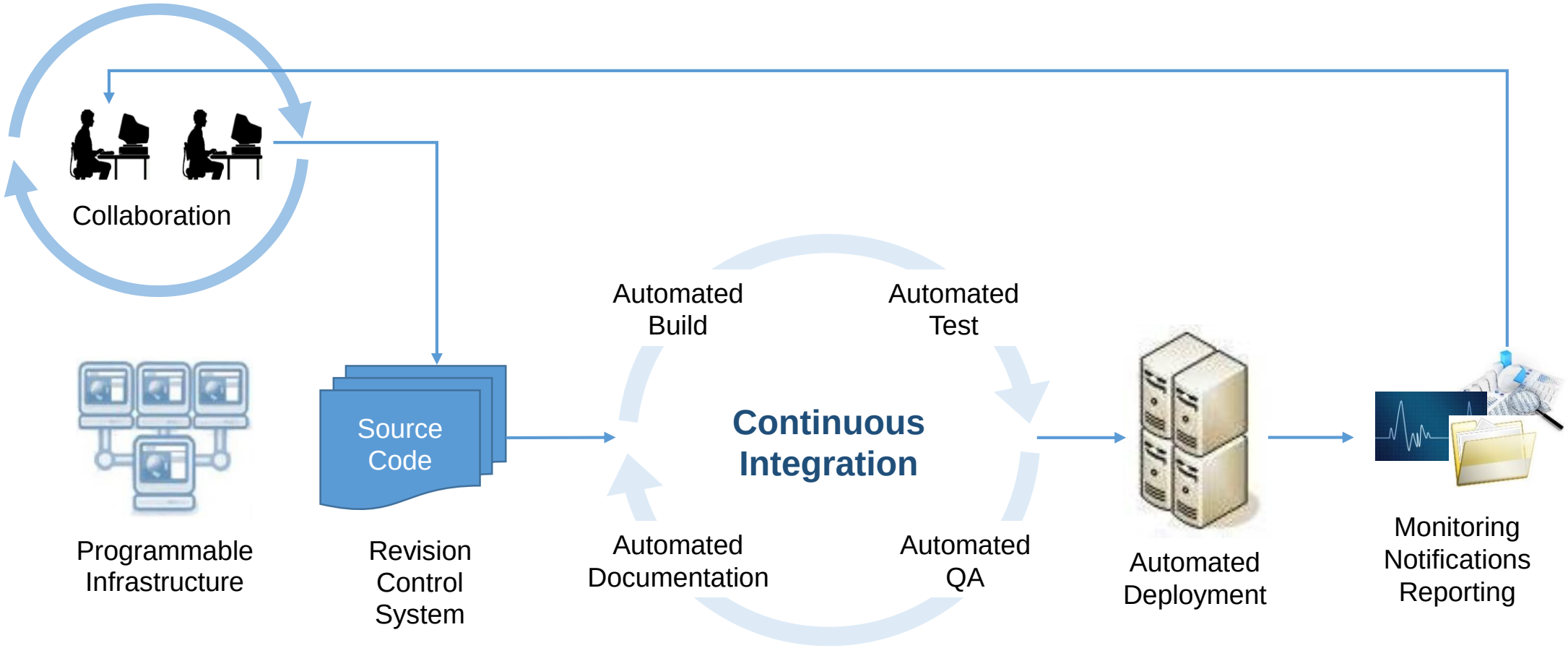
DevOps术语是Development与Operations的合并简写，描述于精简软件交付流程，强调从生产环境到开发的生命周期快速地反馈学习。





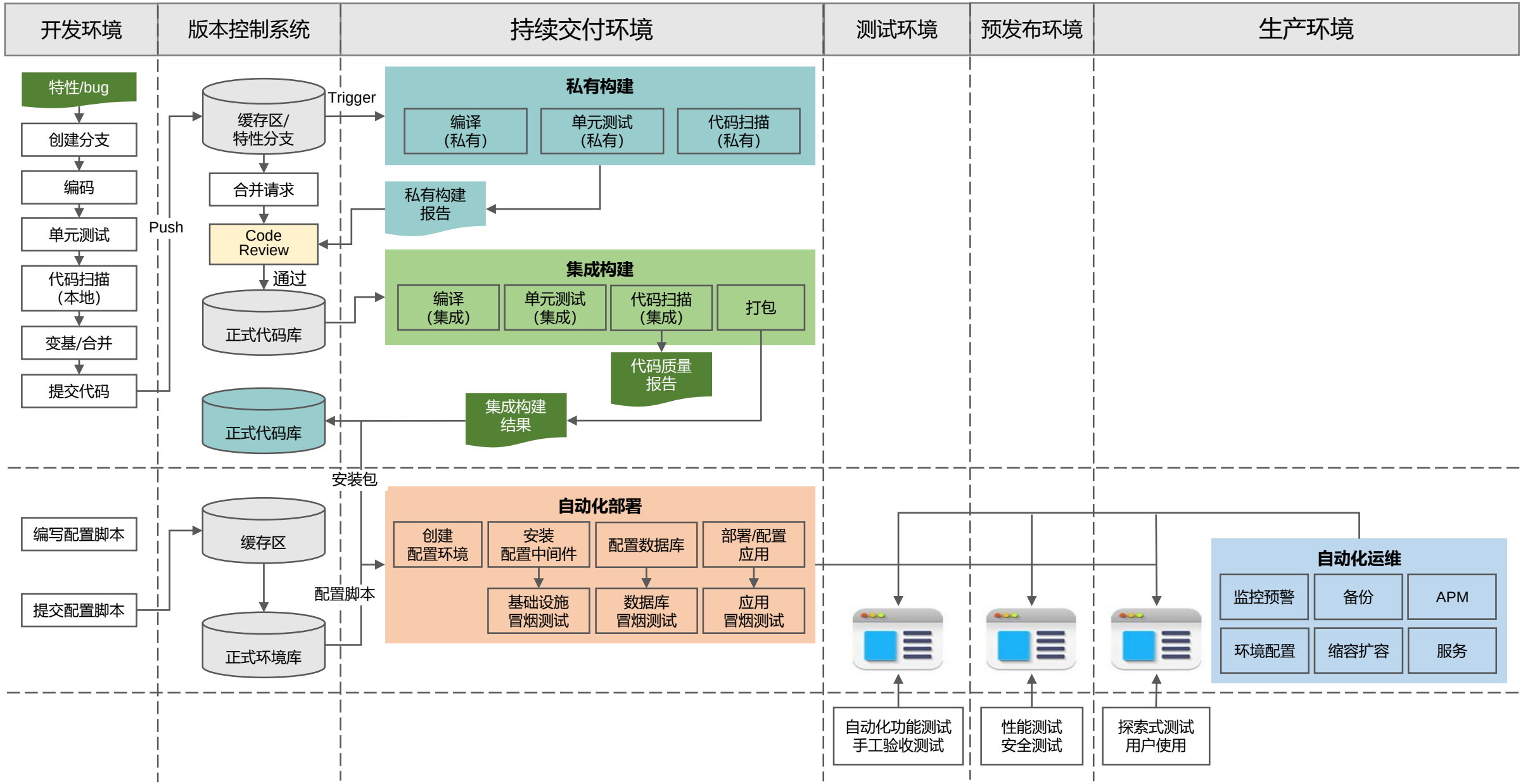


DevOps

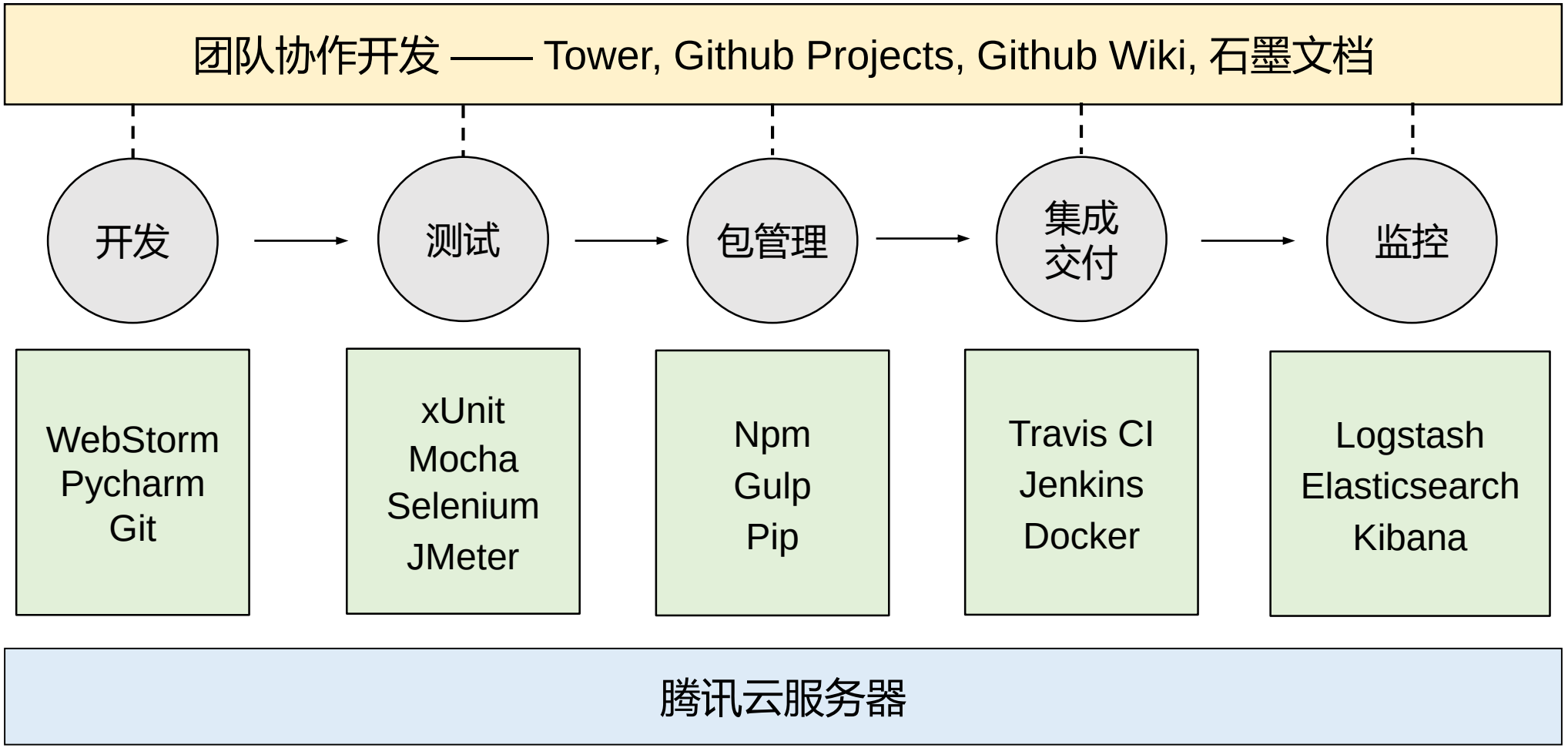

Collaboration
Programmable Infrastructure
Source Code
Revision Control System
Automated Build
Automated Test
Continuous Integration
Automated Documentation
Automated QA
Automated Deployment
Monitoring Notifications Reporting

DevOps

- **编码**：代码开发和审阅，版本控制工具、代码合并工具
- **构建**：持续集成工具、构建状态统计工具
- **测试**：通过测试和结果确定绩效的工具
- **打包**：成品仓库、应用程序部署前暂存
- **发布**：变更管理、发布审批、发布自动化
- **配置**：基础架构配置和部署，基础架构即代码工具
- **监视**：应用程序性能监视、最终用户体验


开发环境
版本控制系统
持续交付环境
测试环境
预发布环境
生产环境
特性/bug
创建分支
编码
单元测试
代码扫描
(本地)
变基/合并
提交代码
Push
缓存区/
特性分支
合并请求
Code
Review
通过
正式代码库
Trigger
私有构建
编译
(私有)
单元测试
(私有)
代码扫描
(私有)
私有构建
报告
集成构建
编译
(集成)
单元测试
(集成)
代码扫描
(集成)
打包
代码质量
报告
集成构建
结果
正式代码库
安装包
编写配置脚本
提交配置脚本
缓存区
正式环境库
配置脚本
自动化部署
创建
配置环境
安装
配置中间件
配置数据库
部署/配置
应用
基础设施
冒烟测试
数据库
冒烟测试
应用
冒烟测试
自动化运维
监控预警
备份
APM
环境配置
缩容扩容
服务
自动化功能测试
手工验收测试
性能测试
安全测试
探索式测试
用户使用

DevOps工具链


团队协作开发 —— Tower, Github Projects, Github Wiki, 石墨文档
开发
测试
包管理
集成
交付
监控
WebStorm
Pycharm
Git
xUnit
Mocha
Selenium
JMeter
Npm
Gulp
Pip
Travis CI
Jenkins
Docker
Logstash
Elasticsearch
Kibana
腾讯云服务器

Travis CI

Travis CI 是在软件开发领域中一个在线的、分布式的持续集成服务，用来构建及测试在GitHub托管的代码。

https://travis-ci.org/

- 目前大多数github项目都已经移入到Travis CI的构建队列中，据说Travis CI每天运行超过4000次完整构建
- 支持多种编程语言，包括Ruby、JavaScript、Java、Scala、PHP、Haskell和Erlang等
- 免费支持Github项目，通过yml语法驱动执行，开通Travis后只需编写.travis.yml就能完成持续集成，简洁清新而且独树一帜

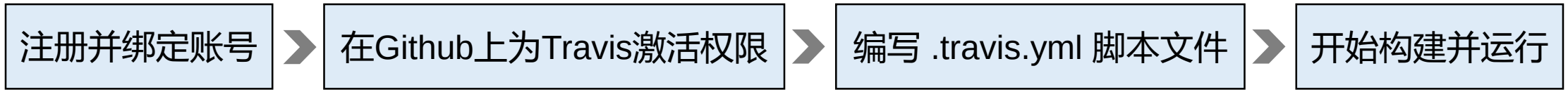
Travis CI

Travis CI 只支持 Github，不支持其他代码托管服务，因此必须满足以下条件：

- 拥有 GitHub 帐号
- 该帐号下面有一个项目
- 该项目里面有可运行的代码
- 该项目还包含构建或测试脚本

官方文档 https://docs.travis-ci.com/
https://www.jianshu.com/p/8e6e4fd2aae7

使用 Travis CI 的基本流程：



Jenkins

Jenkins是一个开源的、提供友好操作界面的持续集成工具，主要用于持续和自动地构建与测试软件项目、监控外部任务的运行。

https://jenkins.io/

易安装：只有一个 java -jar jenkins.war，从官网下载该文件后直接运行，无需额外的安装，更无需安装数据库

易配置：提供友好的GUI配置界面

测试报告：以图表等形式提供详细的JUnit/TestNG测试报表

分布式构建：把集成构建等工作分发到多台计算机中完成

文件识别：跟踪每次构建生成哪些jar包以及使用哪个版本jar包

插件支持：通过第三方插件扩展，使其变得更加强大

谢谢大家！

THANKS