Software Development 2.0

Embracing eco-system and open source

Software Development 2.0

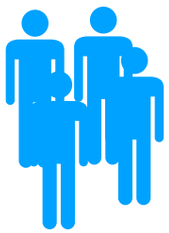
- 内建行业最佳实践
  - 组件化
  - 持续交付
- 不强制具体研发流程
- 拥抱开源与生态系统

Software Development 2.0

Source Control

Package Management

Software Development 2.0

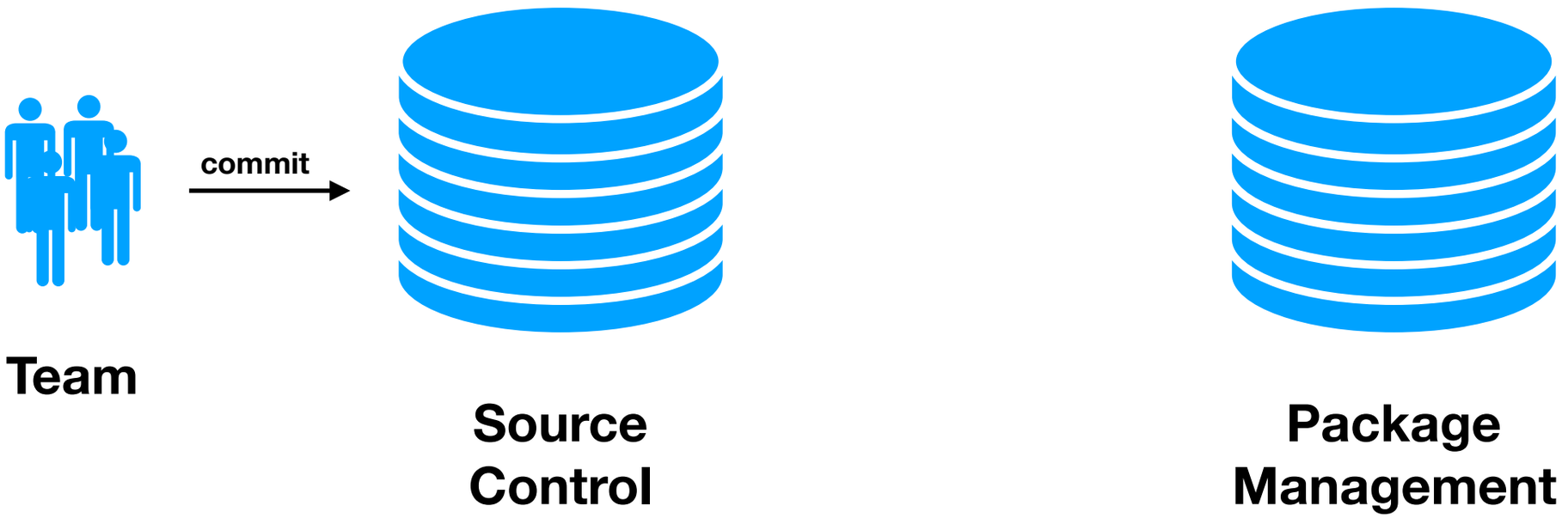


Team

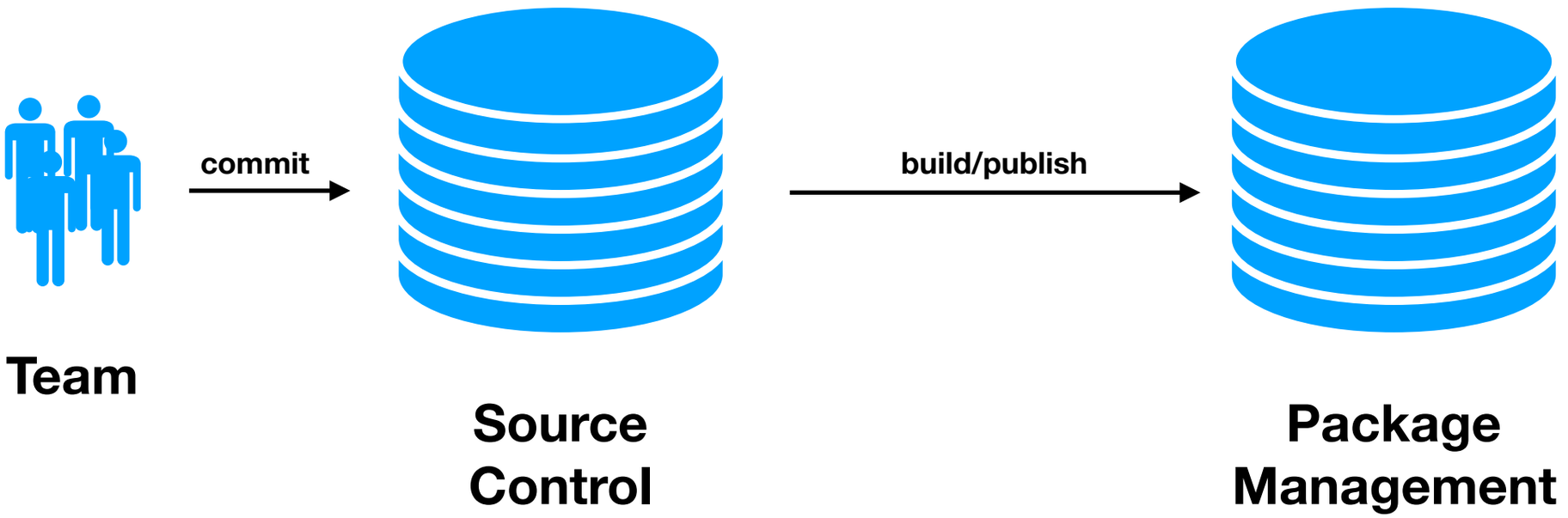
Source Control

Package Management

Software Development 2.0



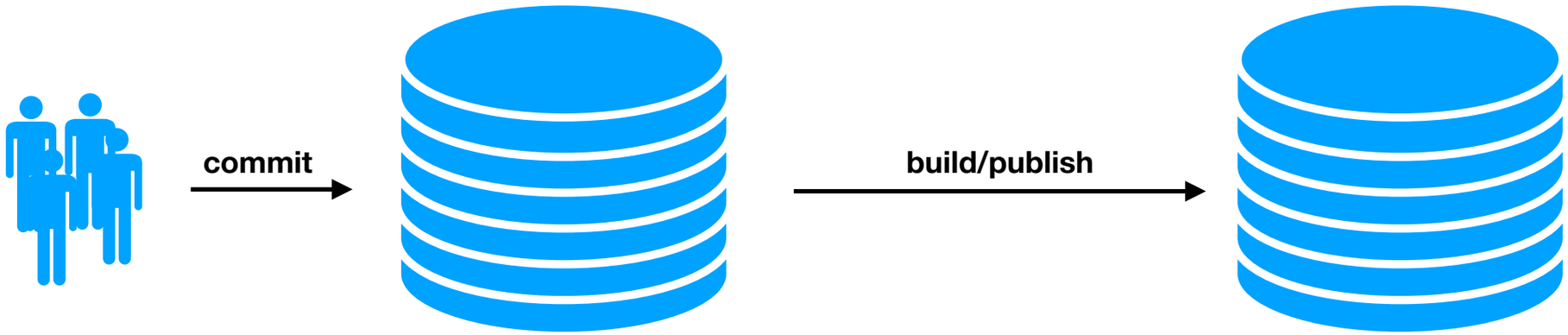
Software Development 2.0



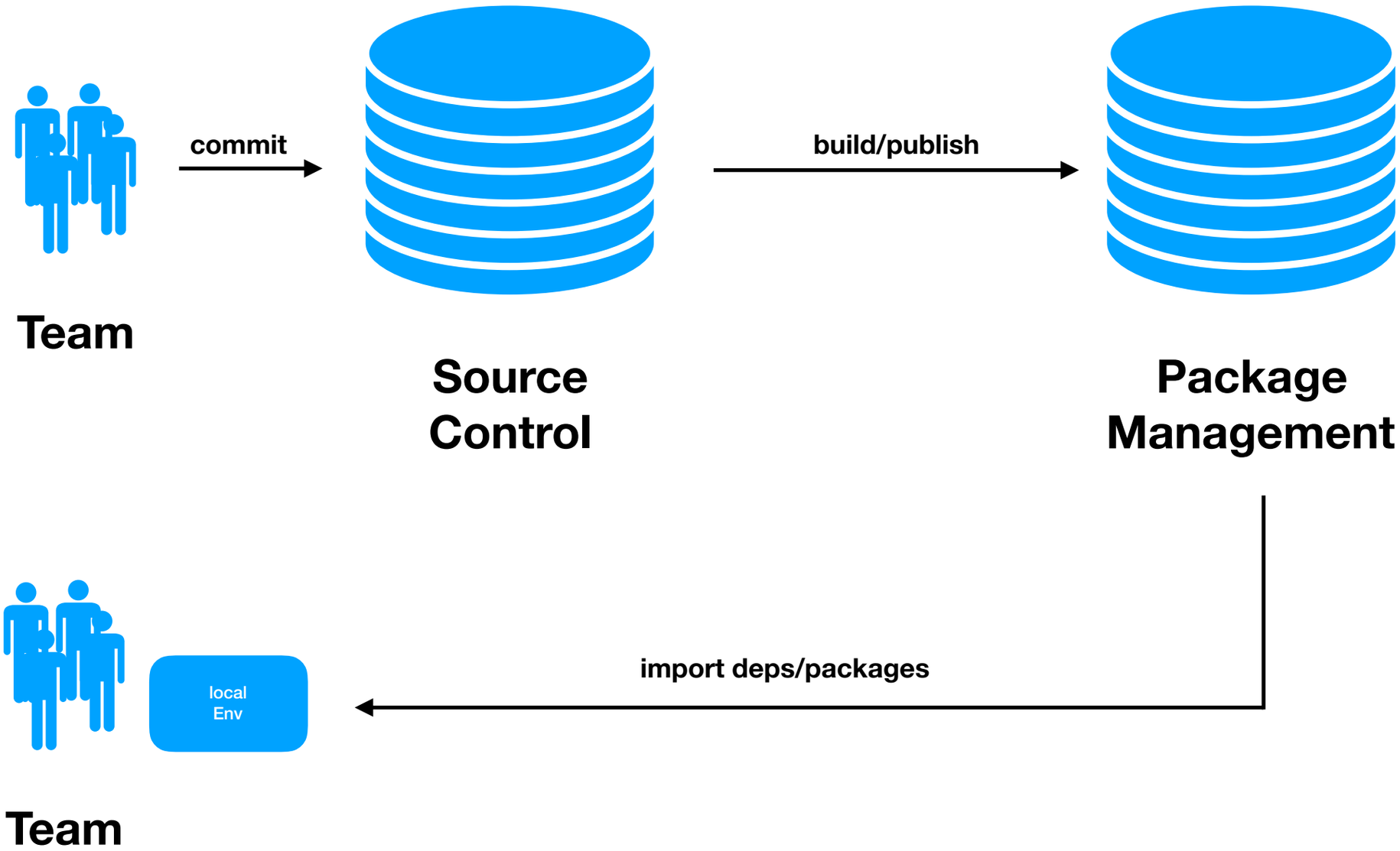
Software Development 2.0

- 软件编码在过去20年中，以源代码管理为终结。代码提交到SCM研发工作在概念上就已经结束
    - 构建/发布 被看作额外的工作
    - 需要额外的动力去构建持续构建和发布流程
- 而通过引入包管理工具，需要在构建工件进入包管理器内，研发才能完成
    - 构建/发布 被内建在流程之内
    - 每个团队必须构建对应的构建流程意以完成开发流程

Software Development 2.0



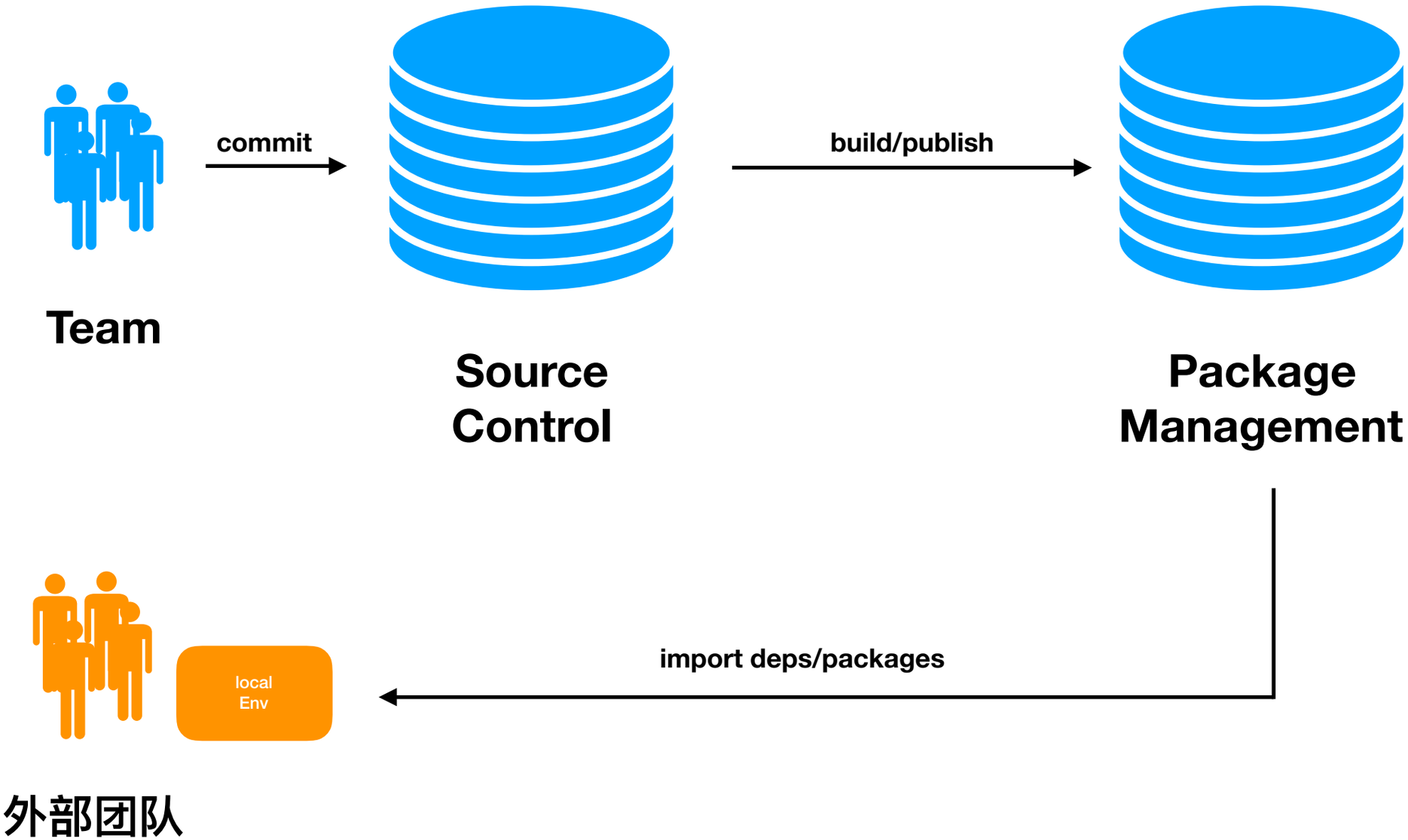
Software Development 2.0



Software Development 2.0

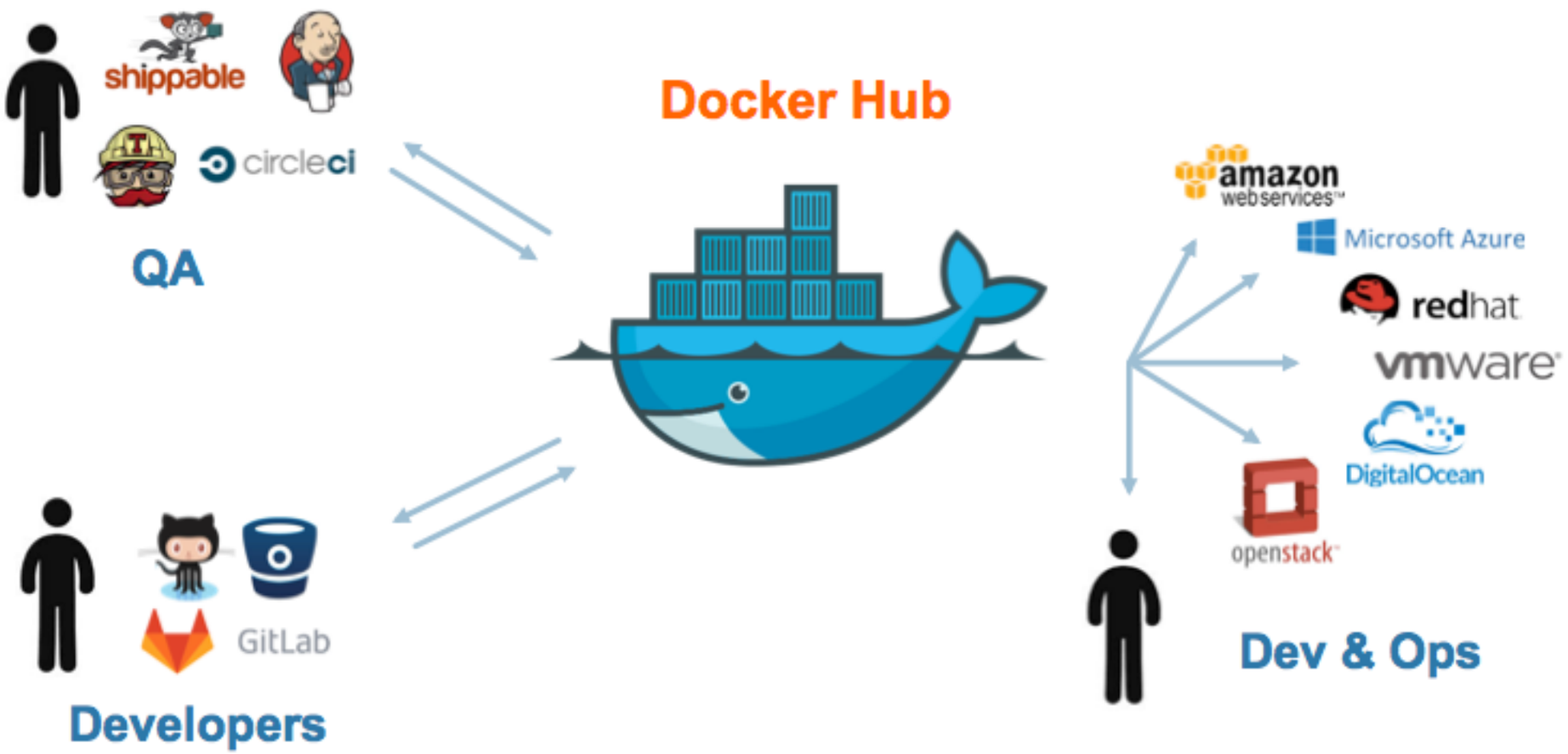
- 软件编码在过去20年中，依赖管理通过源代码依赖实现。项目间直接通过源代码管理工具中的目录实现
    - 缺乏稳定发布的概念。其他团队的工作和修改直接影响其他团队
    - 需要理解其他团队的编译构建流程和工具链
    - 假设为同一项目，并不是不同组件。依赖关系隐含。编译优化有限。
- 项目组间从不通过源代码依赖，而是通过发布的包管理器选择依赖的组件
    - 包管理器中的稳定发布包是其他团队依赖是的基础。源代码仓中的代码为组件团队私有。修改传播不可见
    - 以完整发布包形式，隔离构建流程
    - 团队需要对发布的包提供支持，因而可以完成开发者自运维（DevOps的一种形态）
    - 假设为组件团队，项目需要组合不同组件。依赖关系显示化，编译优化简单（细粒度编译）

Software Development 2.0

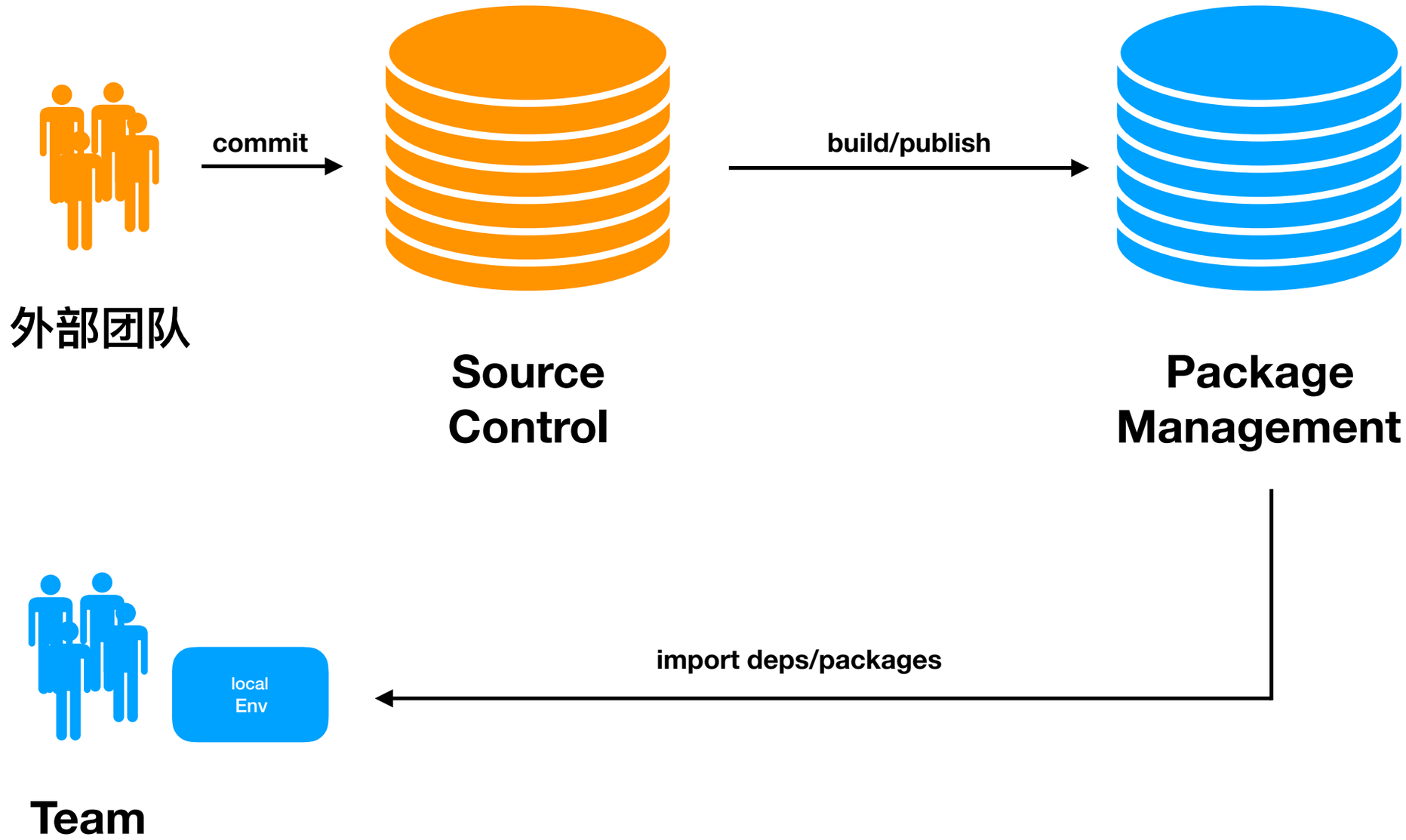


Software Development 2.0

- 统一的包管理器作为分发仓，使得外部团队可以很容易地将组件引入研发全流程
- 围绕统一的分发仓，形成生态系统
  - 开放 （内外团队无区别）
  - 竞争（内外团队无区别）


shippable
circleci
QA
Docker Hub
amazon web services
Microsoft Azure
redhat
vmware
DigitalOcean
openstack
GitLab
Developers
Dev & Ops

Software Development 2.0



Software Development 2.0

- 围绕开放的中央分发仓，形成生态
- 对内，团队间以生态形式合作（基于发布件）
- 外部团队间也以生态形式合作（基于发布件）
- 可以充分自由的拥抱开源