

# 基于容器的 持续交付过程实践

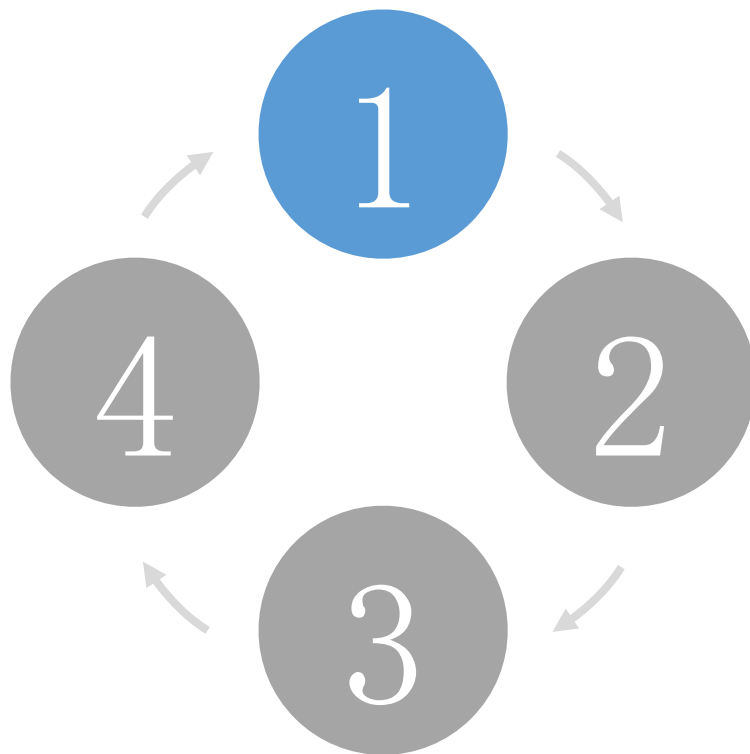
---

中移（苏州）软件技术有限公司

2017.7

04  
应用交付的  
容器化

03  
镜像仓库建  
设与维护



01  
CICD的  
困境与愿景

02  
持续集成  
过程的容器化

# CICD概念与定义

**DevOps:** IT机构分为运营和应用开发。企业正在朝着DevOps的方向发展，将这两个团队结合到一起，2017年应用DevOps的企业从66%上升到74%。

**容器:** “基础设施作为代码”。开发团队应该把环境设置作为代码提供，而不是提供容易出错的安装指令页面。容器在开发中运行的方式与在QA和生产中运行的方式完全相同。

**Jenkins:** 典型的持续集成和交付（CICD）工具，可用作在容器平台本身上运行整个CICD过程的容器



# CI的现状

CI工具主要包括传统CI工具，云计算环境中的CI工具和用于移动应用的CI工具。

场景：传统CI工具

典型：Jenkins

解决：

1. 对源代码控制系统的支持
2. 对依赖管理工具的支持
3. 对各种类型测试的支持

场景：云计算的CI工具

典型：Travis CI

解决：

1. 开发不必关心基础设施和用户认证授权等
2. 提供更多的测试功能：不同浏览器的支持，不同语言的支撑

场景：移动应用的CI工具

典型：Circle CI等

解决：

1. 提供完全不同的依赖管理机制；
2. 自动化工具支撑
3. 发布方式不同

# CD的现状

---

持续集成更强调研发过程的质量控制，持续交付的范围更广，可以认为是：持续集成+自动发布。

## 第一类

节奏慢、版本发布频率不高，上线出故障影响面不广、影响度不高，这类企业对持续集成的需求不会太强烈，应用持续集成的目的更多地是希望开发过程规范化、透明化。

## 第二类

节奏相对慢、版本发布频率不算高，上线出故障影响面广，影响度高，这类企业对持续集成和自动发布都会有需求，应用持续交付的目的是希望把好版本发布的质量关，同时需要在研发过程中控制质量风险，所以需要持续集成。

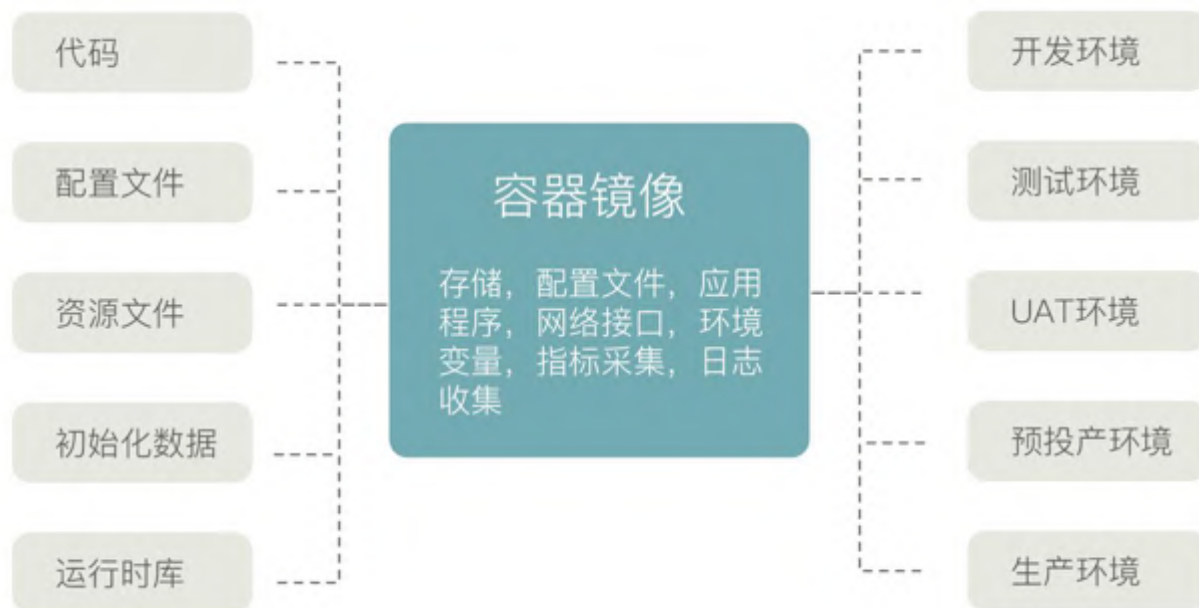
## 第三类

节奏快、版本发布频率高，上线出故障影响面广、影响度高，这类企业一般已经在应用迭代开发、敏捷开发的模式，对持续集成、敏捷测试、自动化测试、自动发布都有强烈的需求。

# CICD的愿景

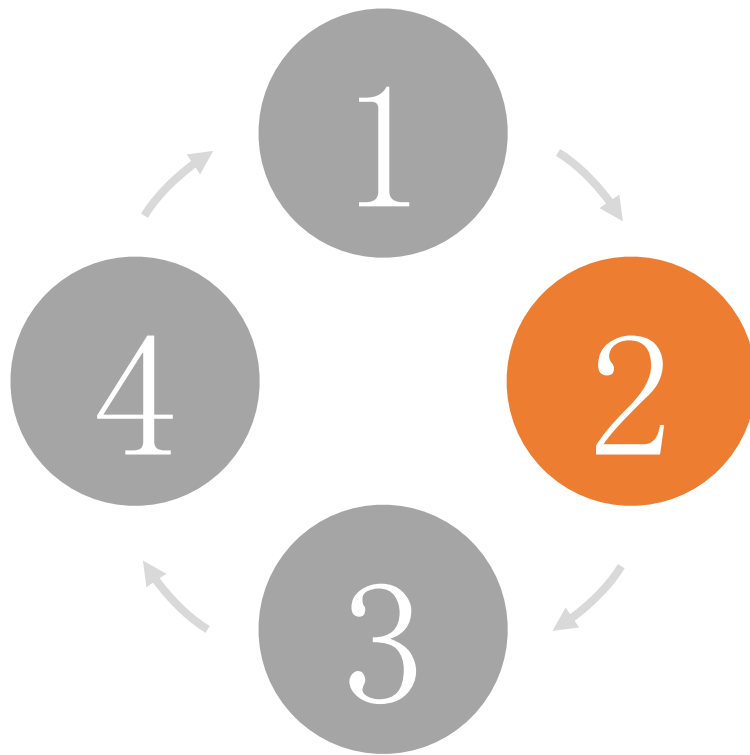
通过建设统一软件商城货架，实现“**一点创新、全网复制**”。架构微服务化，系统容器化。

- 与传统方式相比，通过自动化集成降低人工操作上线风险，满足**研发与运维的无缝衔接**，实现**研运一体**，缩短**50%以上**的上线时间
- 通过制定标准，在开发阶段集成统一的开源日志采集工具，采用自主监控手段，**解决软件维护全外包的问题**



04  
应用交付的  
容器化

03  
镜像仓库建  
设与维护



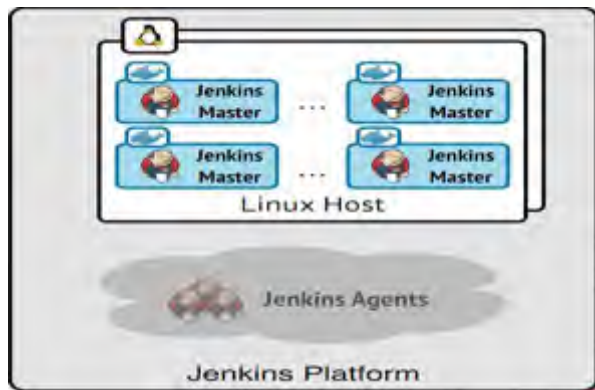
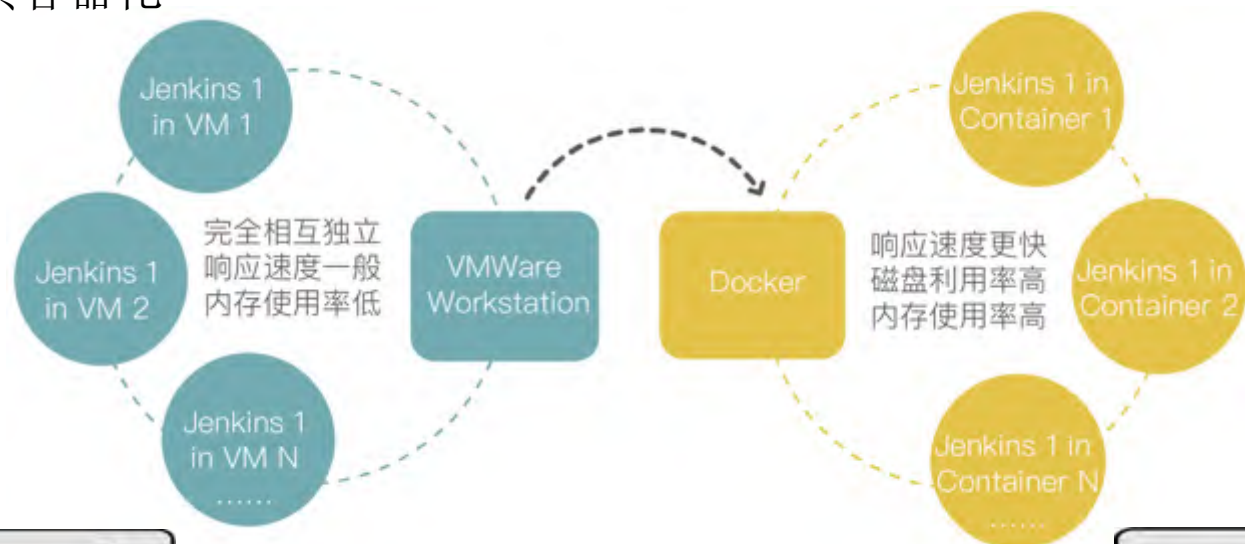
01  
CICD的  
困境与愿景

02  
持续集成  
过程的容器化



# 持续集成过程的容器化—jenkins容器化

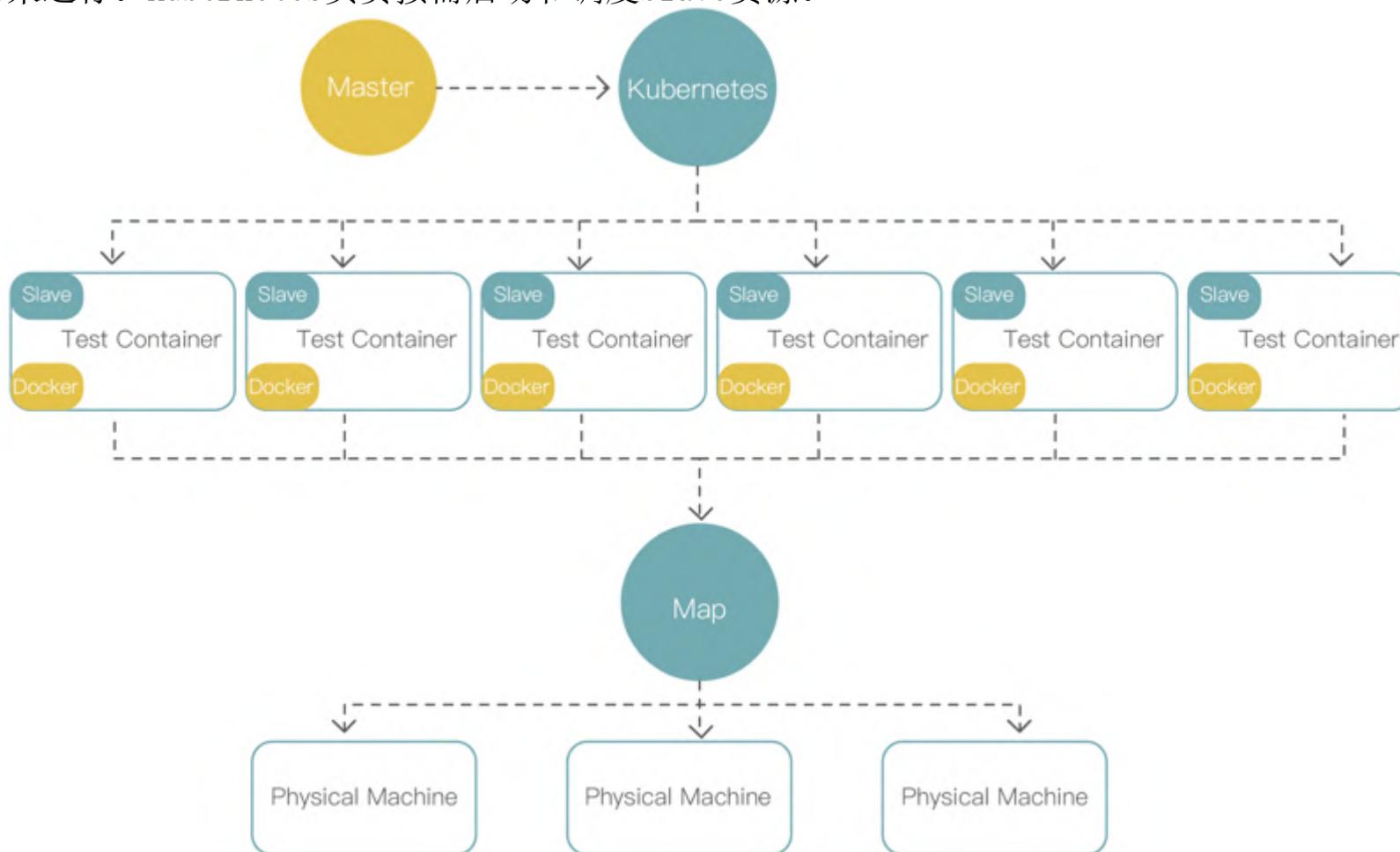
- 困难：产品多，项目多，持续集成环境分散，运维困难
- 方法：持续集成工具容器化



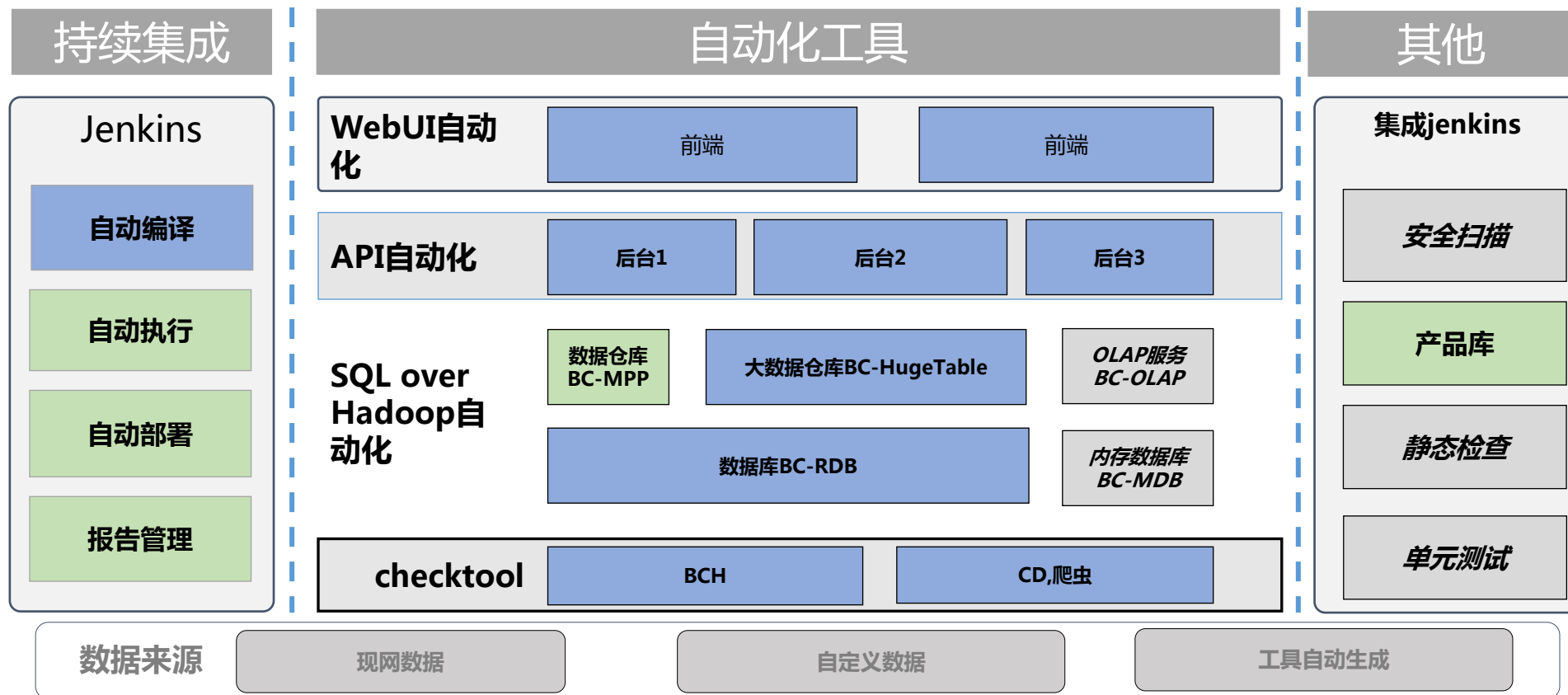


# 持续集成过程的容器化—集群化的持续集成环境

- Master为系统的控制节点，Slave节点提供具体的项目编译测试等工作环境。
- Master使用Docker来运行。Kubernetes负责按需启动和调度Slave资源。

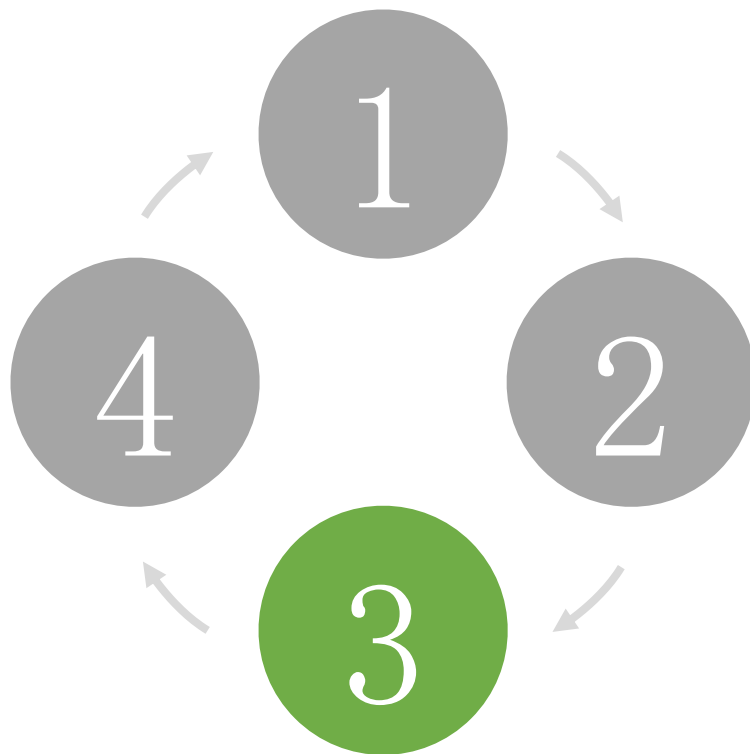


# 持续集成过程的容器化—测试过程自动化



04  
应用交付的  
容器化

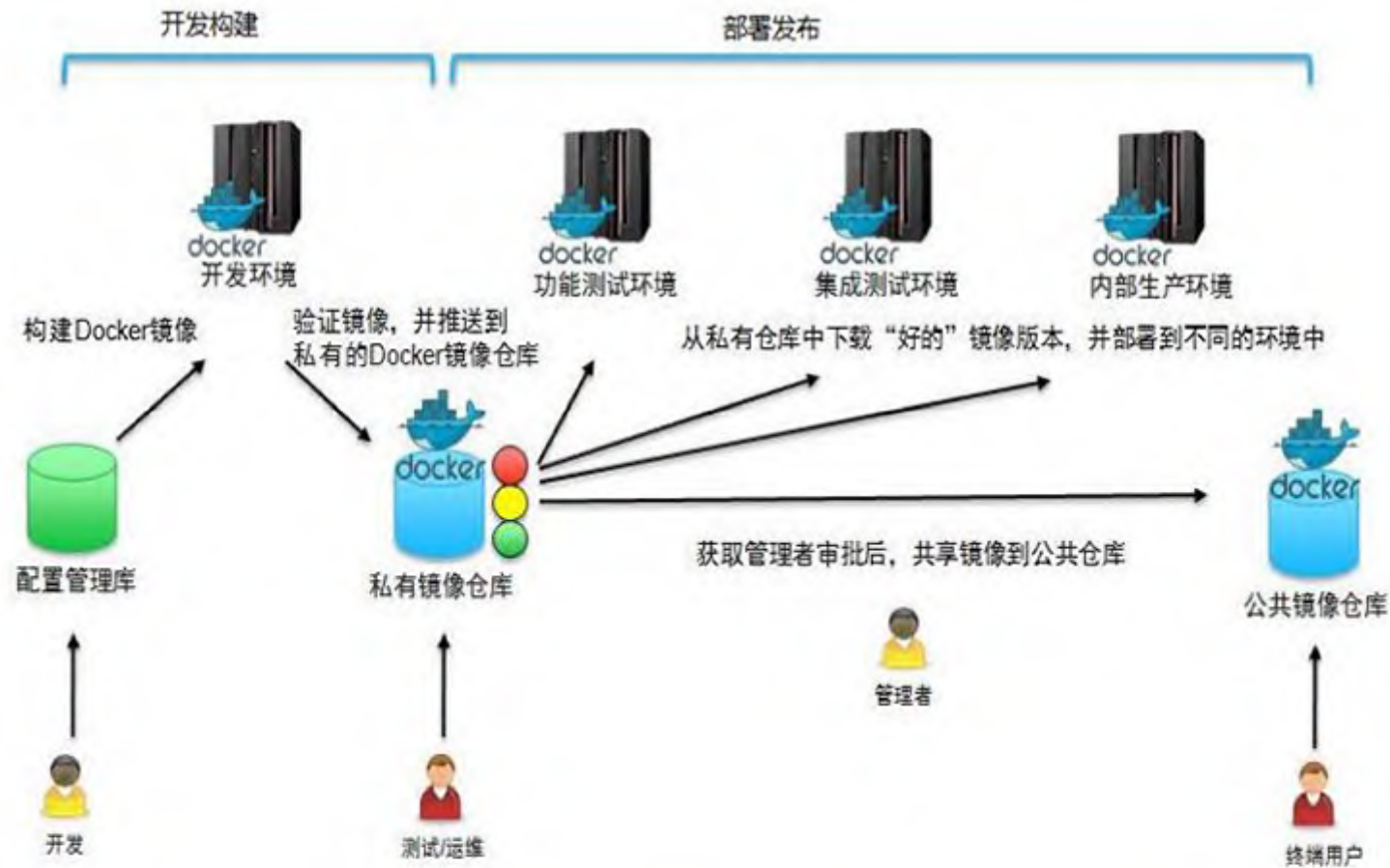
03  
镜像仓库建  
设与维护



01  
CI/CD的  
困境与愿景

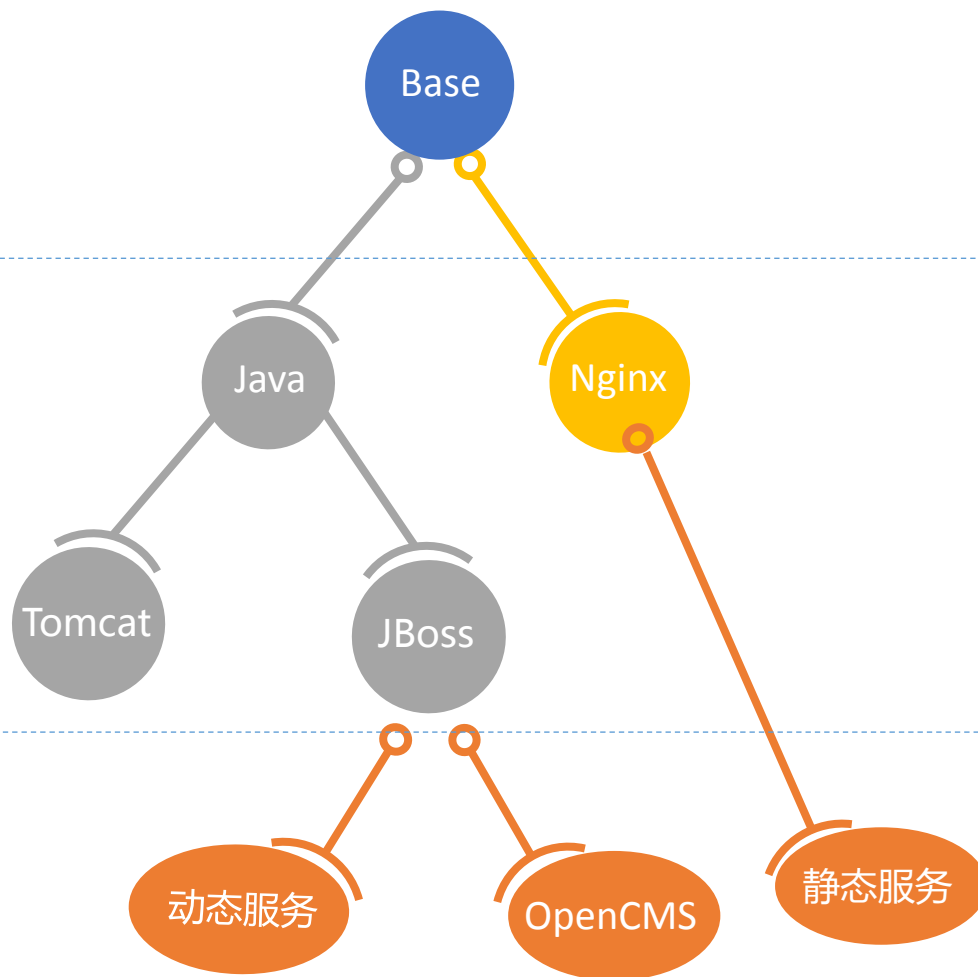
02  
持续集成  
过程的容器化

# 镜像仓库建设与维护—镜像管理



镜像标签使 DevOps 管道下游的使用者可以方便的选择最新最符合要求的镜像

# 镜像仓库建设与维护—镜像分层



## 基础镜像层

安装所有容器都需要的软件

## 服务镜像层

根据能力不同安装不同的软件

## 应用镜像层

仅安装应用程序相关的软件

# 镜像仓库建设与维护—镜像管理

- Harbor和Docker Registry所提供的容器镜像仓库，就是容器镜像的存储和分发服务。提供分层传输机制，优化网络传输；提供web界面，优化用户体验；支持水平扩展；用户管理和集群同步；Harbor支持与k8s的集成。
- 关于镜像传输的流程

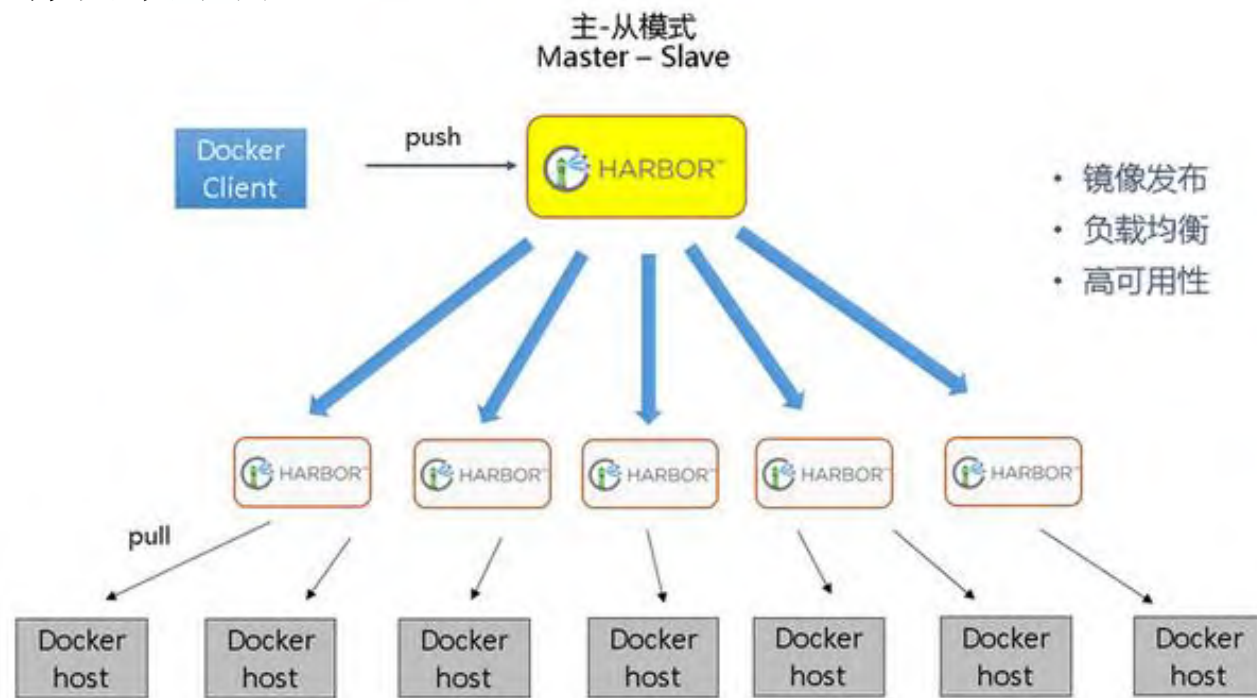




# 镜像仓库建设与维护—引入Harbor

## ● 大规模镜像发布方式

在实际生产运维的中，往往需要把镜像发布到更多的集群节点上。这时，单个Registry已经无法满足大量节点的下载需求，因此要配置多个registry实例做负载均衡。手工维护多个registry实例上的镜像，将是十分繁琐的事情。Harbor支持一主多从的镜像发布模式，可以解决大规模镜像发布的难题。



# 镜像仓库建设与维护—开发测试流程的改变

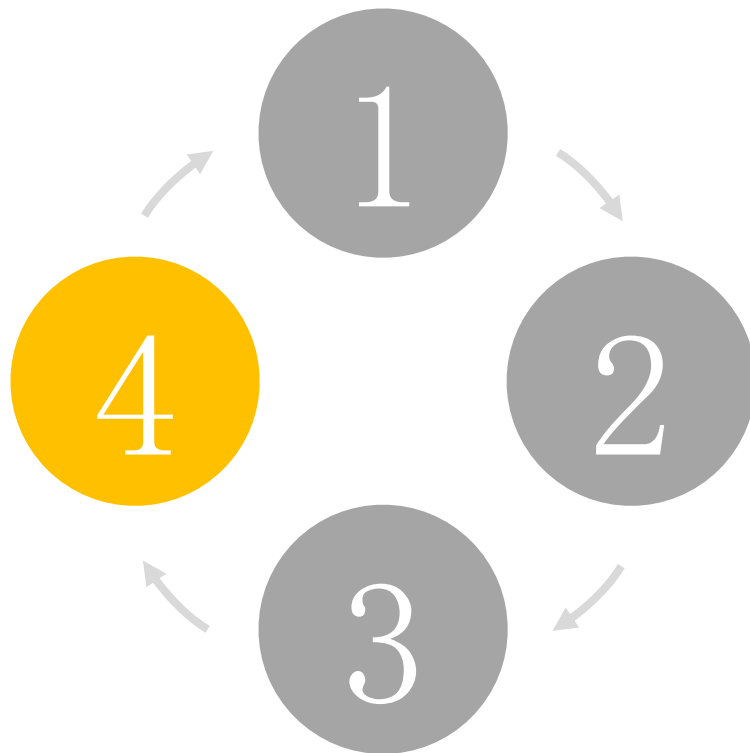


04

应用交付的  
容器化

03

镜像仓库建  
设与维护



01

CICD的  
困境与愿景

02

持续集成  
过程的容器化

# 应用交付的容器化—传统业务上线对比

- **缺点：** 环节多，沟通成本大；涉及部门多，费时费力；时间周期长



## 统一持续集成环境

- 统一持续集成环境的迁移
- 统一的maven仓库
- 统一的分支策略
- 定期的配置检查与审计

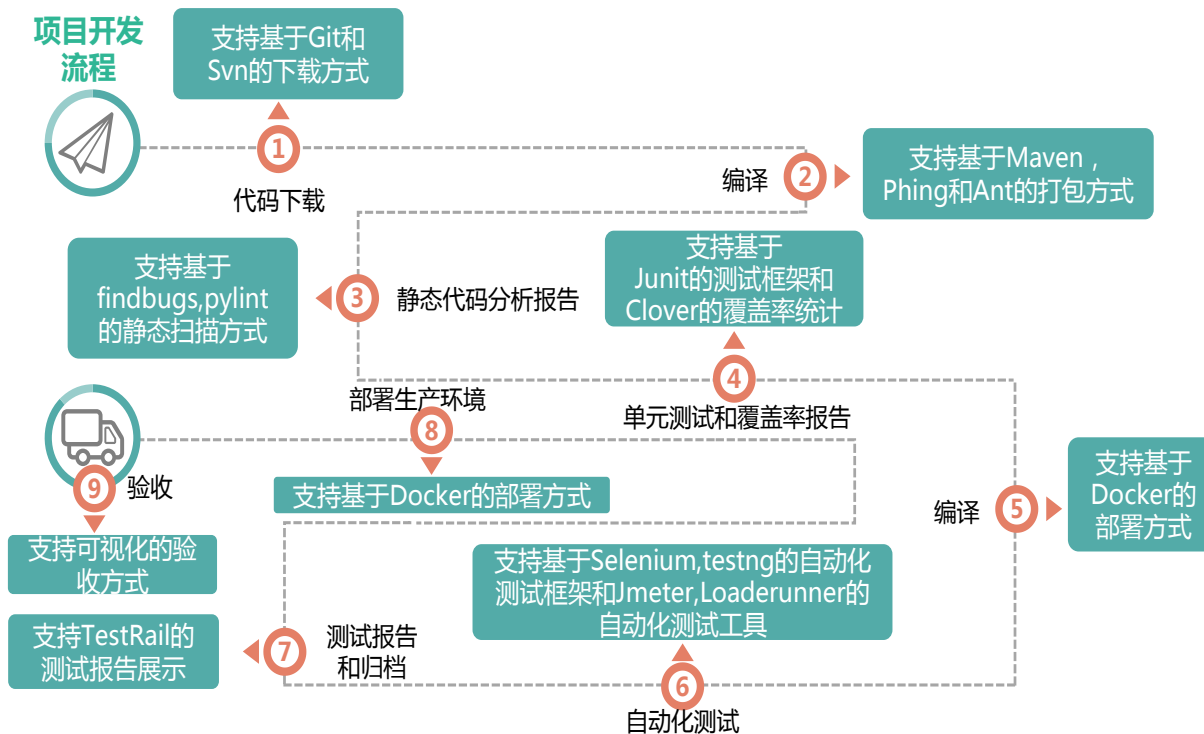
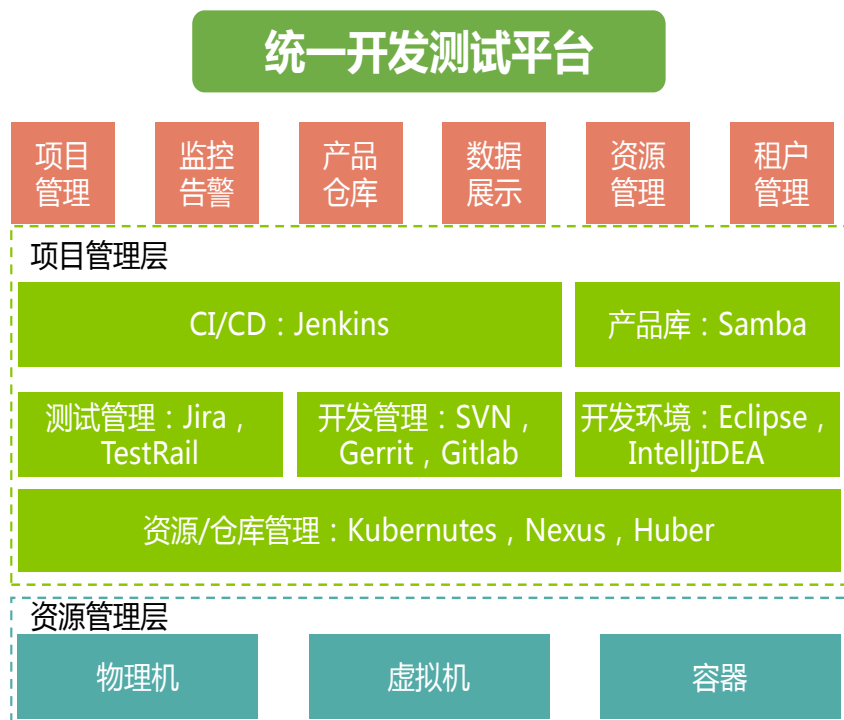
## 打造持续集成流水线

- 适配多语言静态检查
- 建设统一的镜像仓库
- 架构一致的自动化接口测试
- 架构一致的自动化功能测试
- 部署自动化

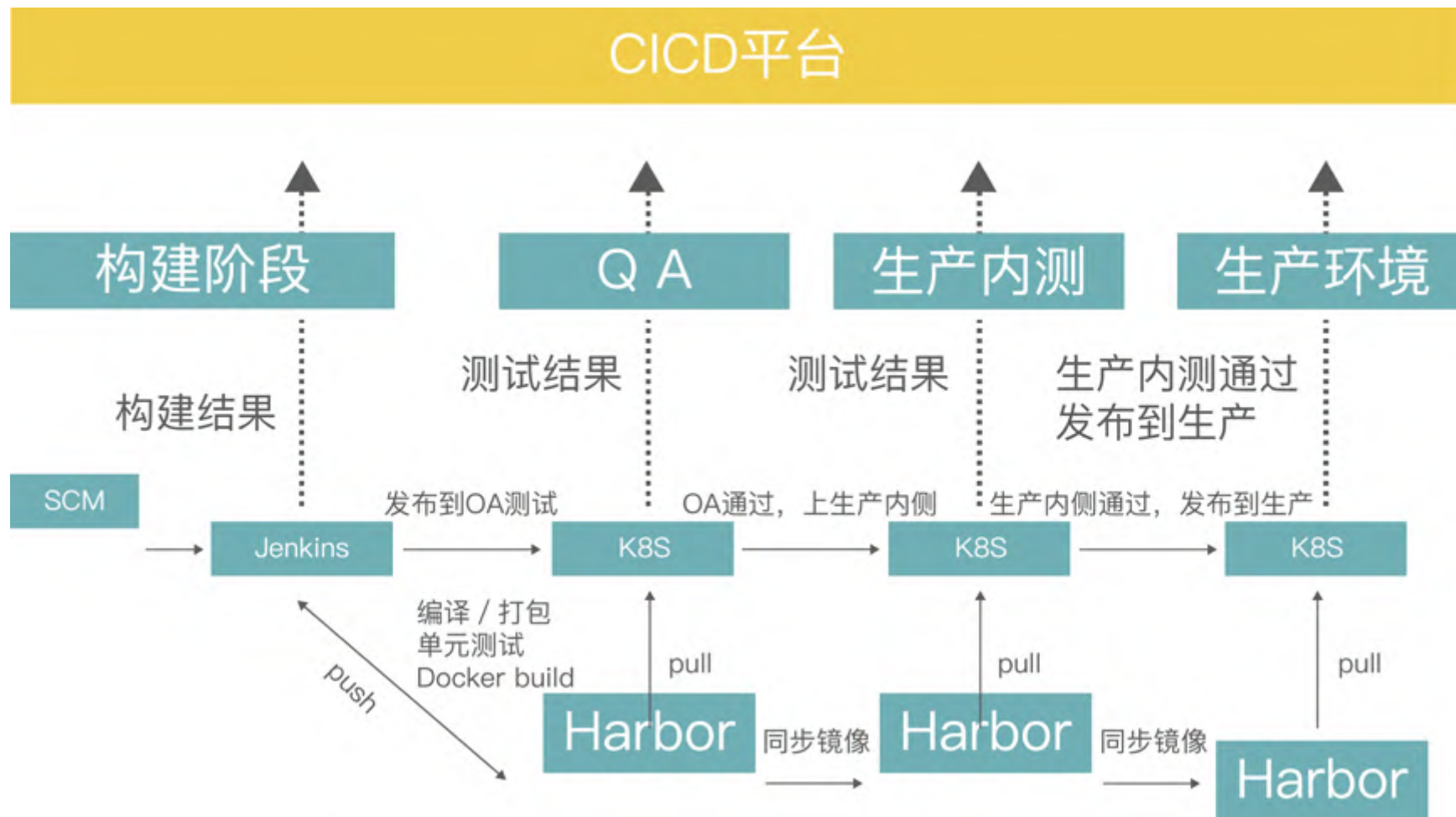
## 多维度的质量视图

- 交付人员工作量视图
- 产品质量视图
- 交付流程视图
- 产品及项目库视图

# 应用交付的容器化—CI/CD平台架构

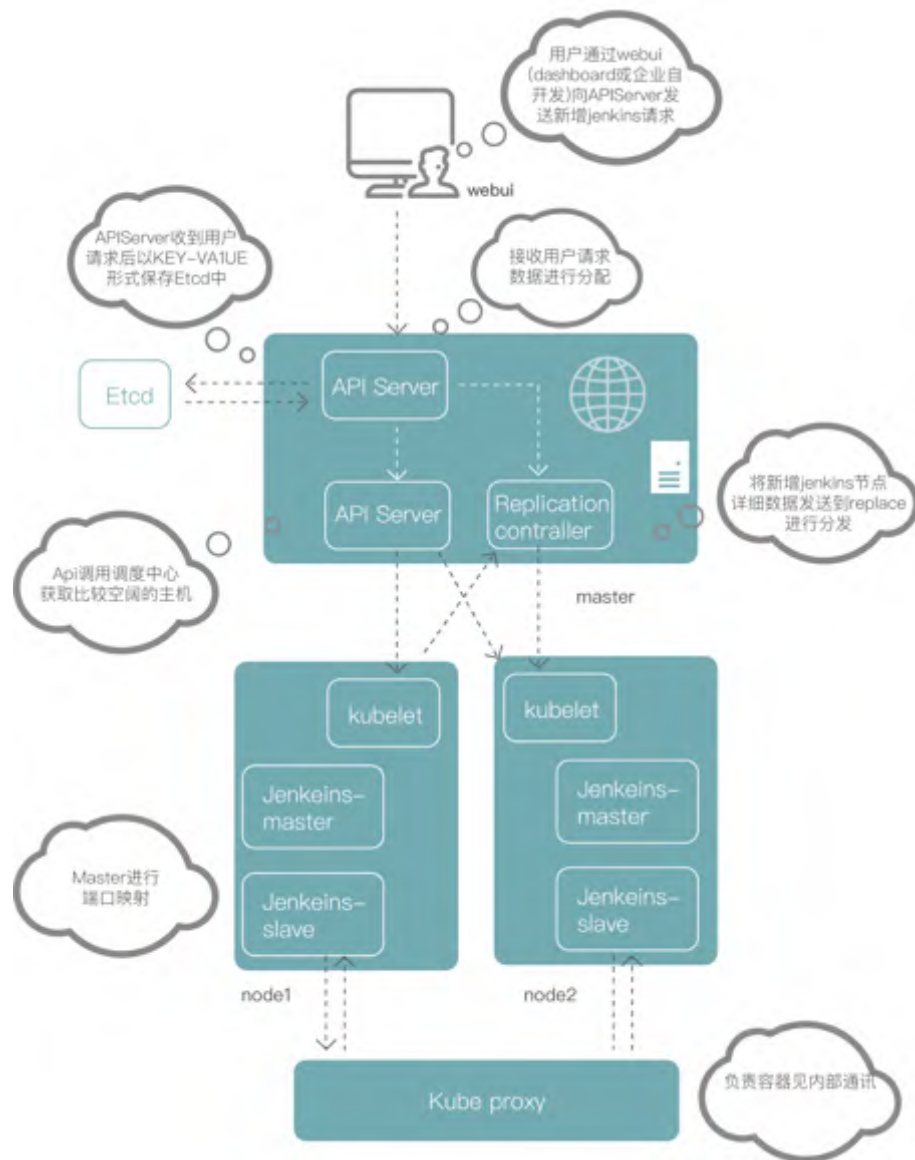


# 应用交付的容器化—引入k8s资源调度

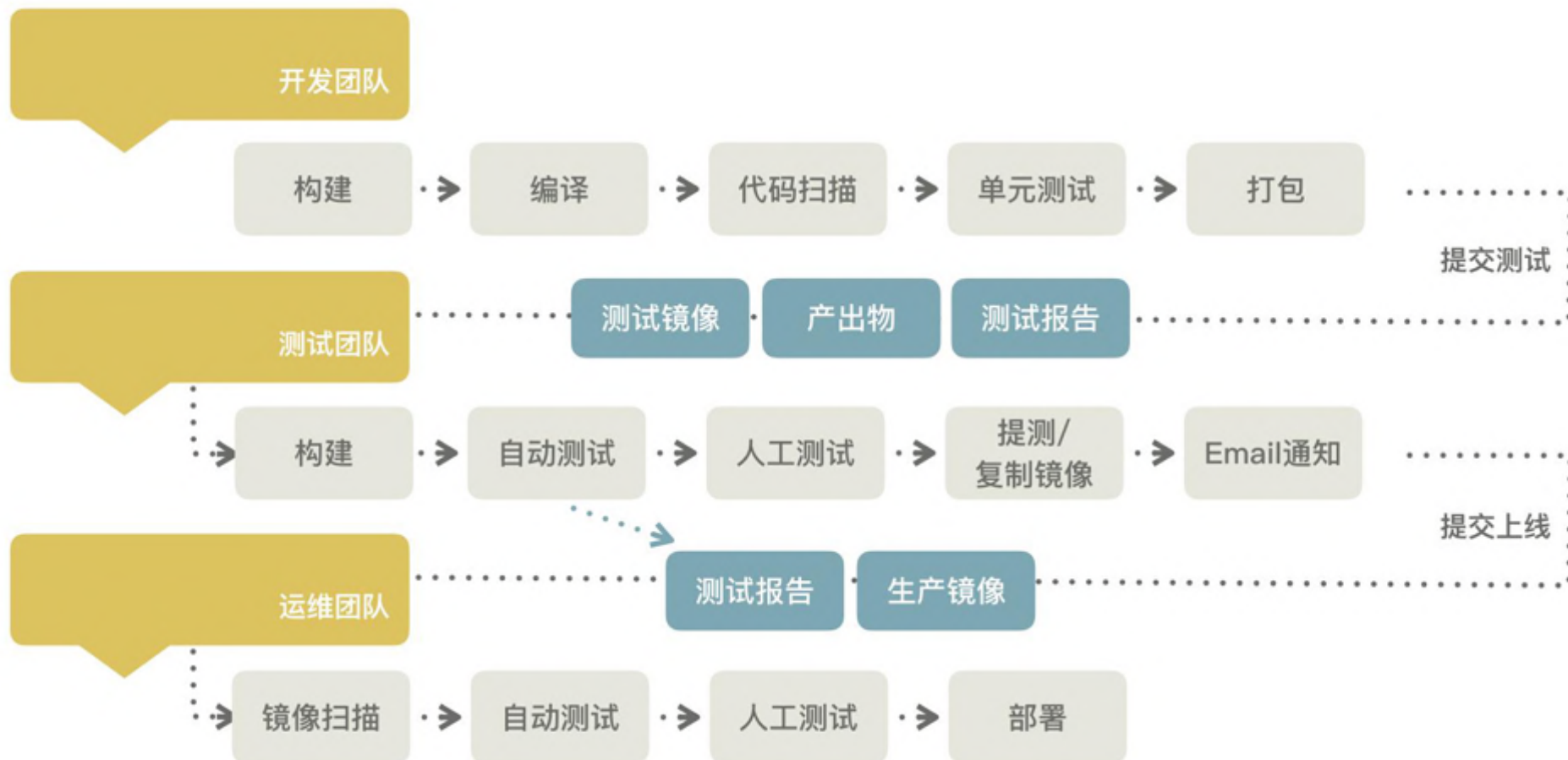




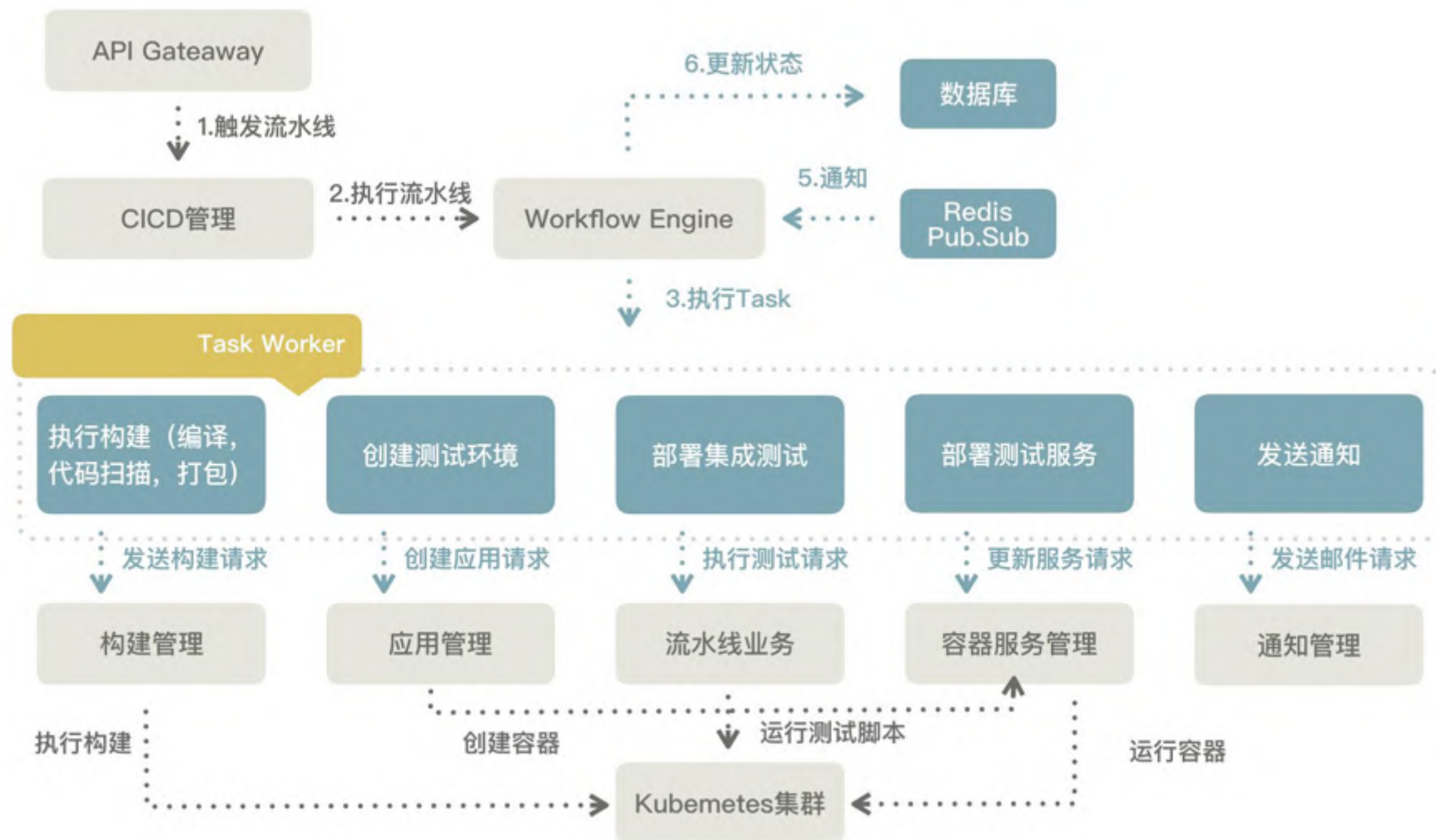
# 应用交付的容器化—k8s资源调度技术实现



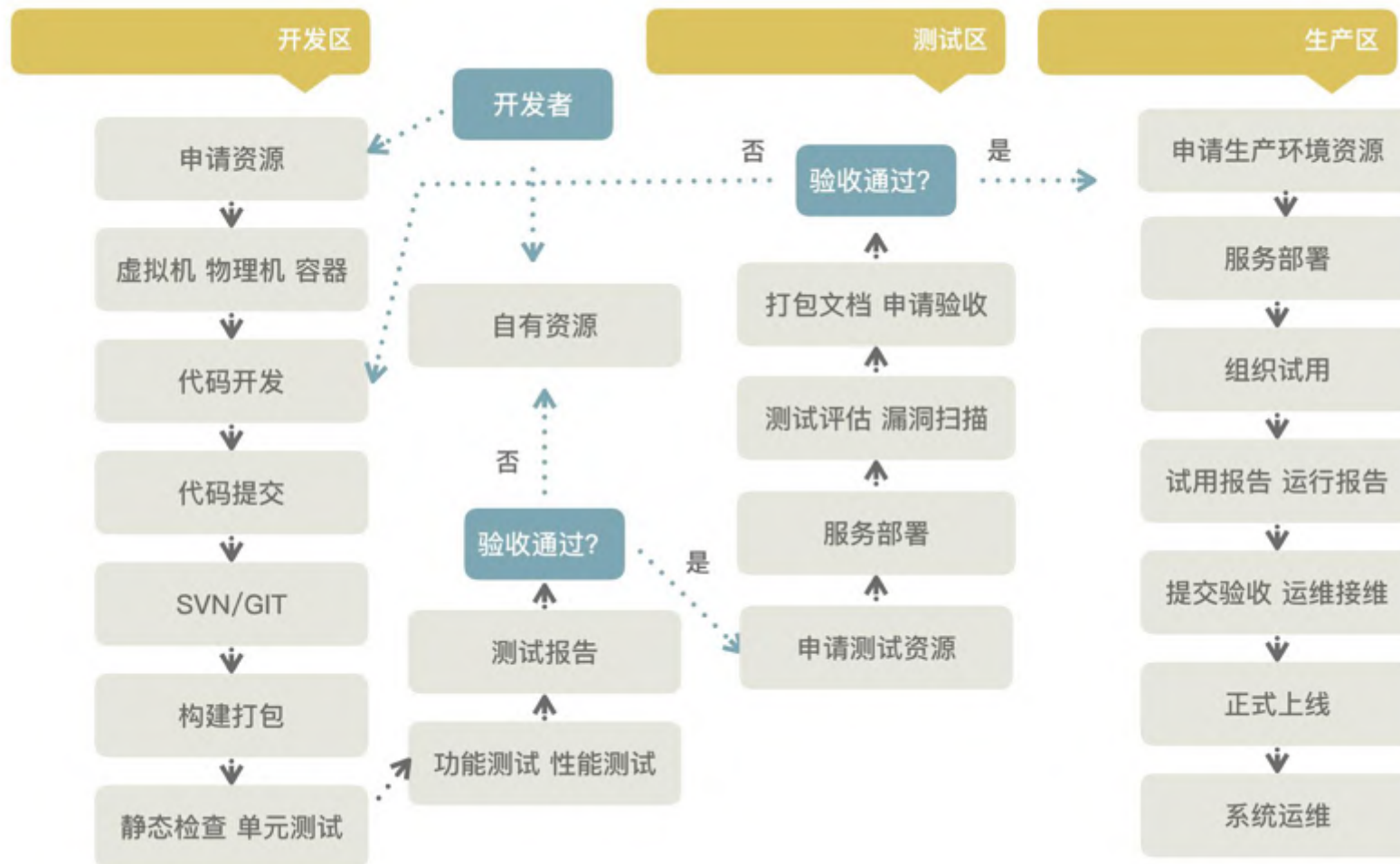
# 应用交付的容器化：业务流程



# 应用交付的容器化：执行过程



# 应用交付的容器化：角色的业务逻辑



# 应用交付的容器化—部署流水线

- 困难：交付进展不清晰，需要对接各种工具链
- 方法：引入Jenkins 2.0 Pipeline 建设可视化的产品一键式发布





# 应用交付的容器化—CI/CD平台





# 应用交付的容器化—CI/CD平台

The screenshot shows the '敏捷开发平台' (Agile Development Platform) interface. The top navigation bar includes '平台总览', '开发管理', '测试管理', '数据分析', and '产品仓库'. The main content area is titled '贵州移动网络代维管理平台'. On the left, there is a sidebar with '流程类' (Process Class) and '代码仓库类' (Code Repository Class). The main workspace contains a pipeline configuration with a toolbar with buttons: '+ 移动', '连接', '更新', '清空', and '运行'. Below the toolbar is a visual pipeline flow: start → svn → java → FindBugs → UnitTest → 测试环境 → 自动化测试 → 发布上线 → 生成报告 → harbor → endNode.

The screenshot shows the '敏捷开发平台' (Agile Development Platform) interface. The top navigation bar includes '平台总览', '项目管理', '持续集成', '数据分析', and '产品仓库'. The main content area is titled '海南大数据一期'. On the left, there is a sidebar with '流程类' (Process Class), '代码仓库类' (Code Repository Class), and '开发语言类' (Development Language Class). The main workspace shows a pipeline execution progress bar with buttons: '返回编辑' and '停止'. The progress bar displays five stages: '代码检出' (2017-07-17 13:24:26, 耗时: 2秒), '代码编译' (2017-07-17 13:24:28, 耗时: 4秒), '静态检测' (2017-07-17 13:24:32, 耗时: 3秒), '单元测试' (等待执行...), and '一键部署' (等待执行...). Below the progress bar is a terminal window showing the output of the 'FindBugs' stage:

```
[Pipeline] stage (Static Code Analysis)
Java version: 1.8.0_111, vendor: Oracle Corporation
Java home: /usr/lib/jvm/java-8-openjdk-amd64/jre
OS name: "linux", version: "2.6.32-431.el6.x86_64", arch: "amd64", family: "unix"
[INFO] Scanning for projects...
[INFO] Building se 1.1.0
[INFO] Fork Value is true
[.java] Warnings generated: 31
```

**THANK YOU !**

