DaoCloud

DaoCloud IT大咖说 知识分享平台

微服务眼中持续交付的最佳姿势

持续交付企业的创新

王天青

首席架构师， DaoCloud云原生转型实验室

About Me

- 南京大学计算机科学与技术系硕士。
- 资深Java程序员，2003年开始从事J2EE开发，从软件开发做到架构设计。
- 08年加入EMC中国研究院，最高担任云平台主任研究员，长期从事云计算创新技术解决方案设计和实现。
- 2015年9月加入麻袋理财，任首席架构师
- 2016年12月加入DaoCloud，任首席架构师，负责云原生应用转型实验室
- 在工作期间，分别在中国和美国提交了20+专利申请，其中7项专利被美国专利局正式授权。

内容摘要

互联网时代的挑战

我要支持**100万**客户！！！

这个功能，我要**明天**上线！！！

又快又好

双十一怎么又**500**了？

公司IT**投入**怎么比业务发展快？？？

独角兽的成功秘诀

- Speed of innovation （天下武功，唯快不破）
- Always-available services （随时、随地可用）
- Web scale（从0到1，快速扩展）
- Mobile-centric user experiences （移动为王）

技术演进

瀑布
Waterfall

桌面电脑
Desktop

单体
Monolithic

物理机
Physical

互联网
IDC

存储
Data Store

敏捷
Agile

移动终端Mobile

分层
N-Tier

虚拟机
Virtual

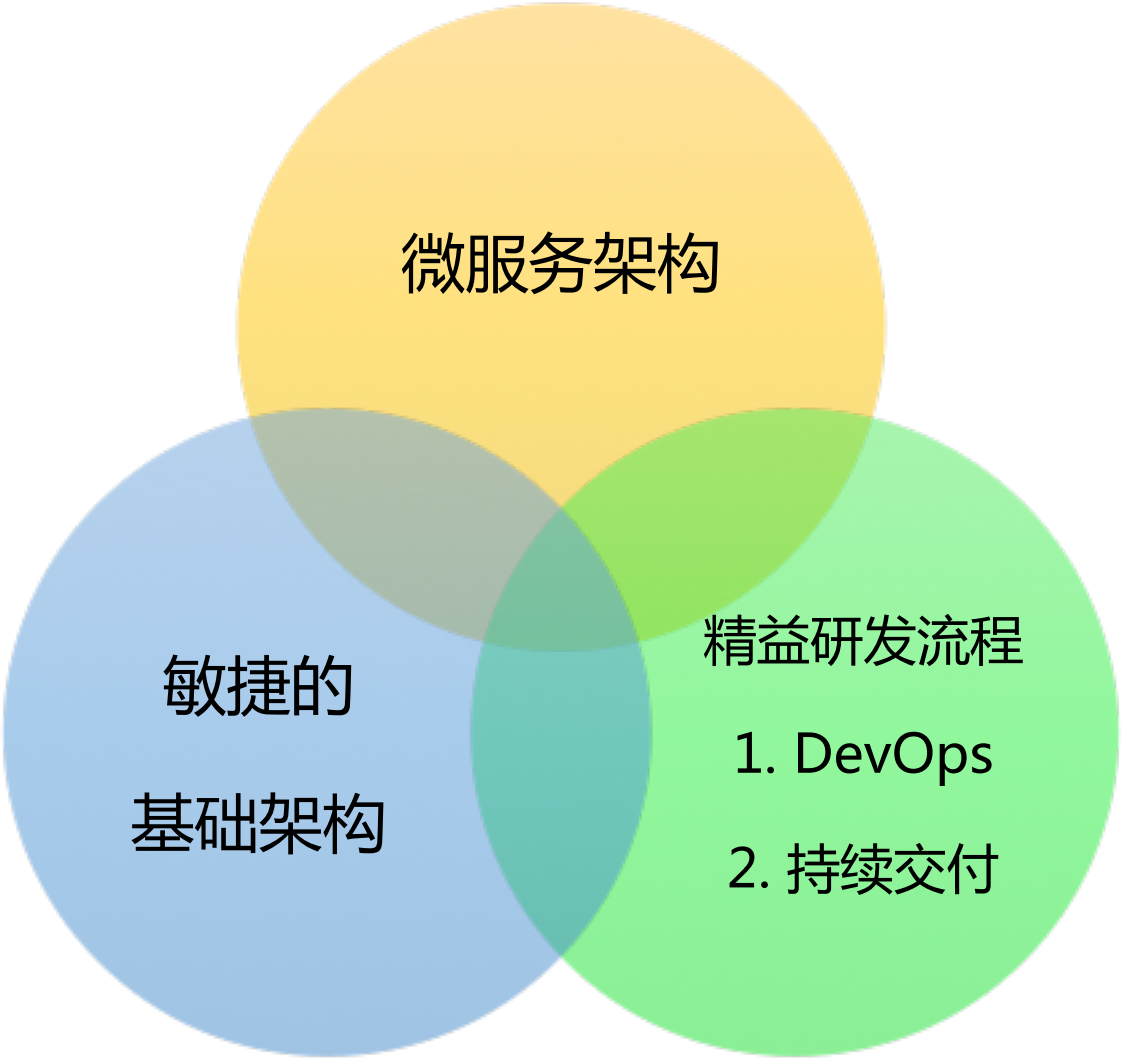
云
Cloud

挖掘
Data Warehouse

解决方案 - 云原生应用

- **Cloud Native Application**
  - *Cloud Native describes the patterns of high performing organizations **delivering** software **faster**, **consistently** and **reliably** at scale.（又快又好）*
  - ***Why**, **how** and **what** of the cloud natives :*
    - ***Continuous delivery***
    - ***DevOps***
    - ***Microservices***

黄金三角



MICROSERVICES

DaoCloud

IT大咖说 知识分享平台

BEFORE

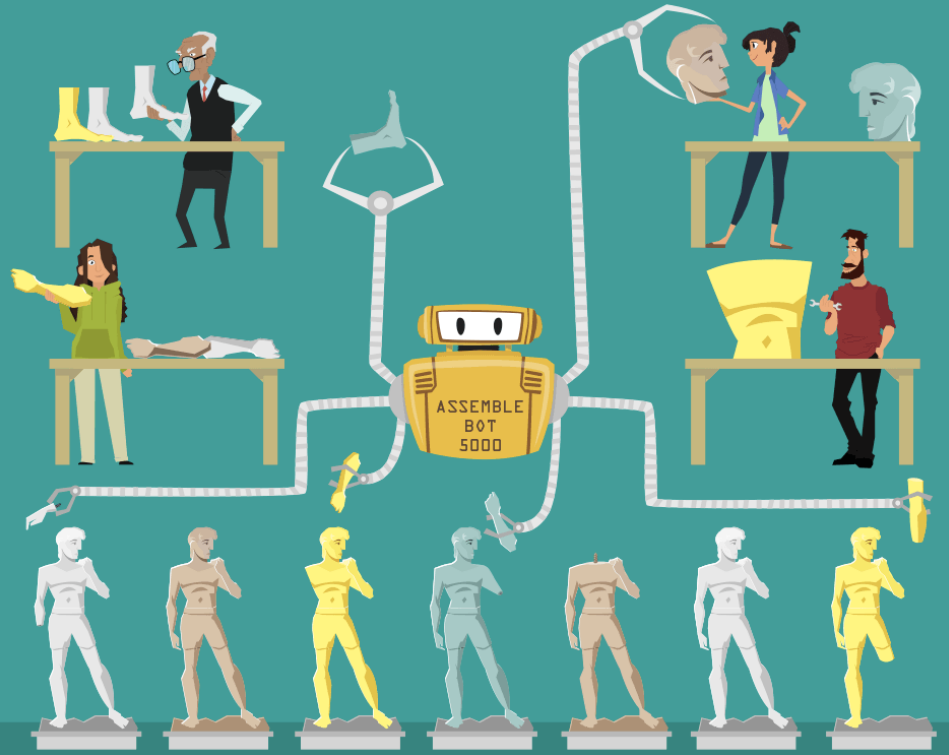
Tightly coupled components

Slow deploy cycles waiting on integrated tests and teams

AFTER

Loosely coupled components

Automated deploy without waiting on individual components



微服务架构定义

- 使用**分而治之**的方式，将一个复杂系统分解为一组微服务
- 一个微服务只服务于某个特定的业务服务（**Bounded Context**）
- 每个微服务可独立运行在自己的进程里
- 一系列独立运行的微服务共同构建起整个系统（**分布式系统**）
- 微服务之间通过一些**轻量**的通信机制进行通信
- 可以使用不同的语言与数据存储技术
- 全自动的部署 / 监控 / 运维机制

DEVOPS

DaoCloud

IT大咖说 知识分享平台

BEFORE

Not my problem

Separate tools, varied incentives, opaque process

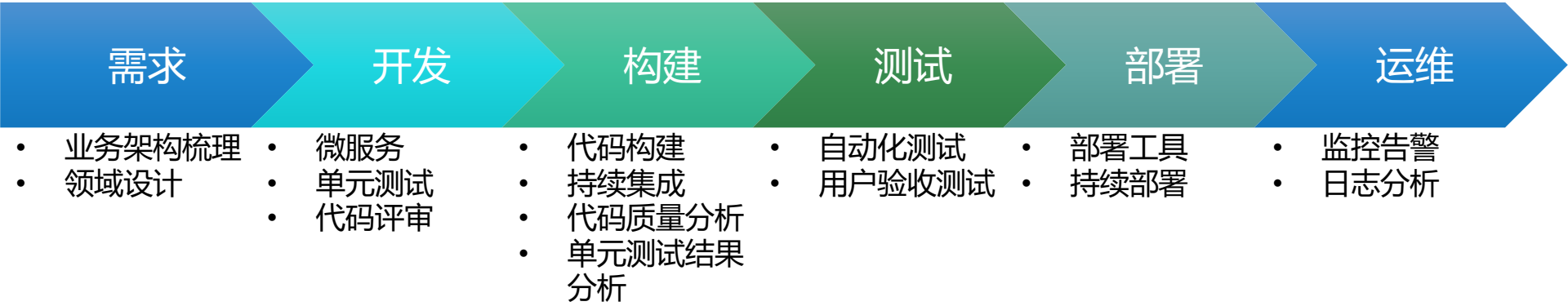
AFTER

Shared responsibility

Common incentives, tools, process and culture

DevOps

- DevOps是一套实践方法，在保证高质量的前提下缩短系统变更从提交到部署至生产环境的时间 – 《DevOps：软件架构师行动指南》
- Ten deploys per day, why not? - 《凤凰项目》



CONTINUOUS DELIVERY

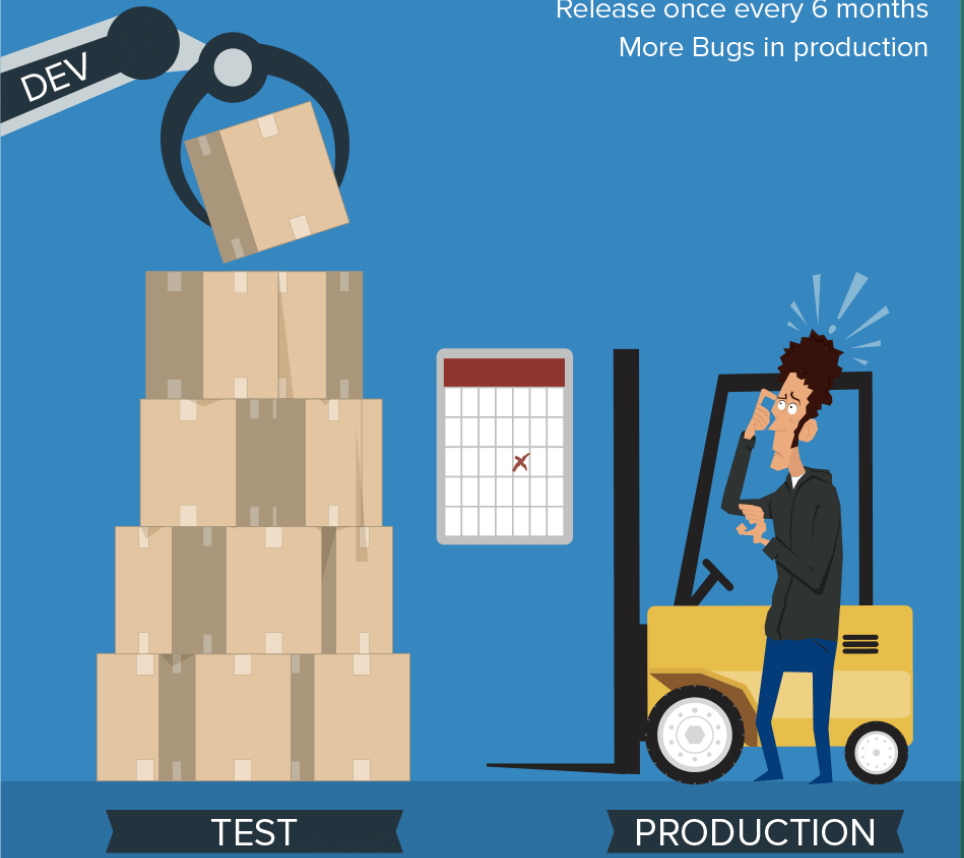
DaoCloud

IT大咖说 知识分享平台

BEFORE

Release once every 6 months

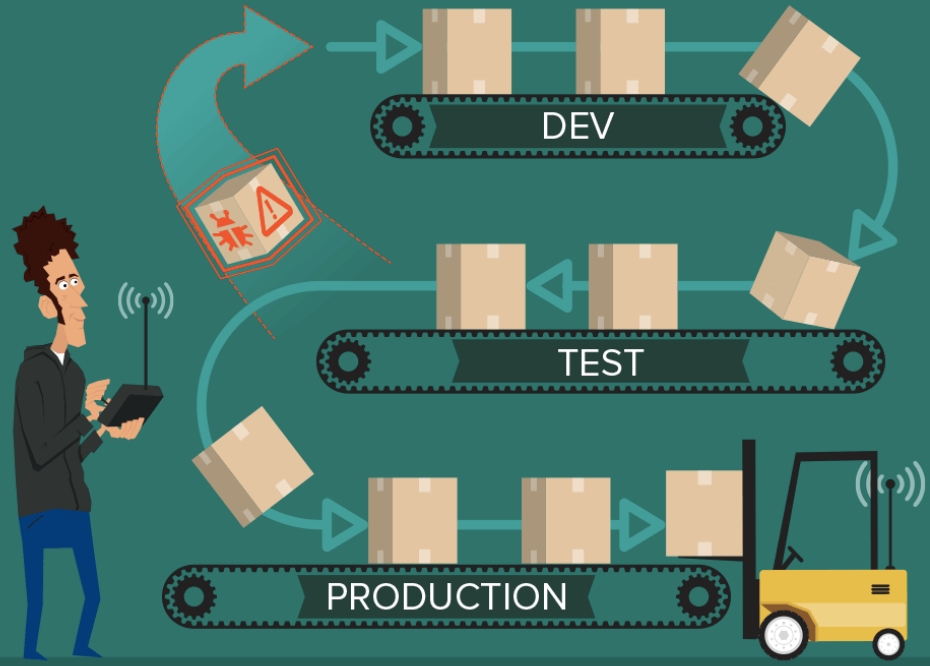
More Bugs in production



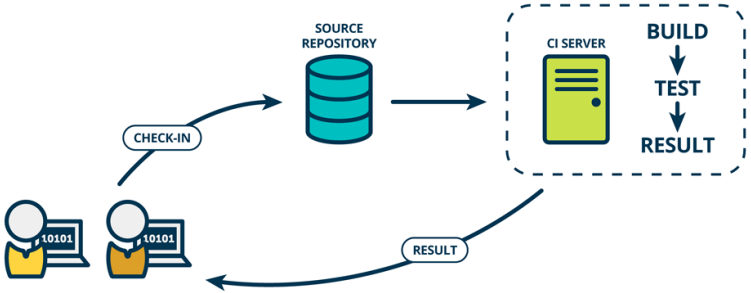
AFTER

Release once a week

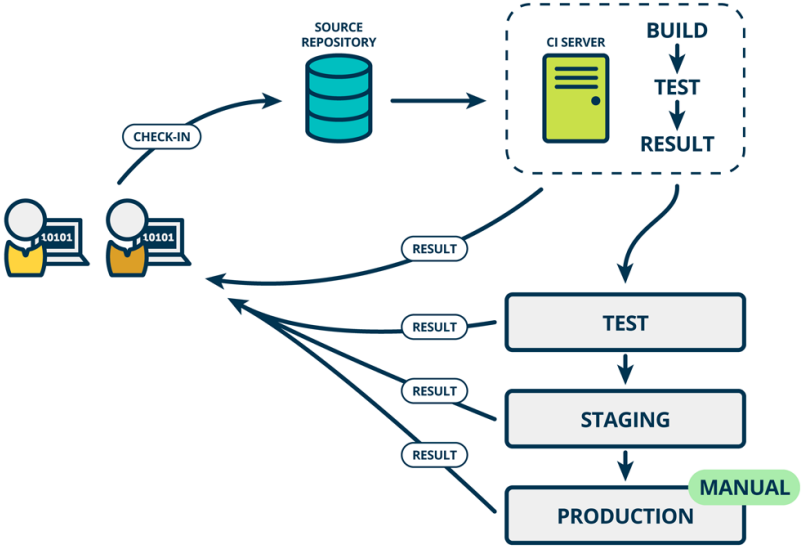
Higher Quality of Code



持续集成 / 持续部署 / 持续交付



持续集成



持续部署

CONTAINERS

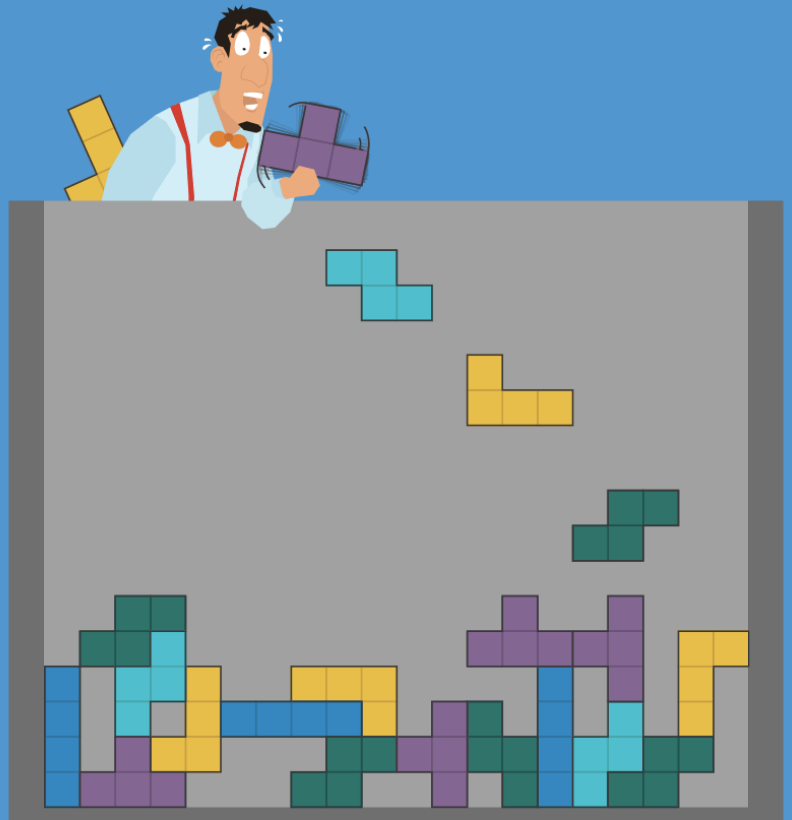
DaoCloud

IT大咖说 知识分享平台

BEFORE

Make sure software matches with hardware

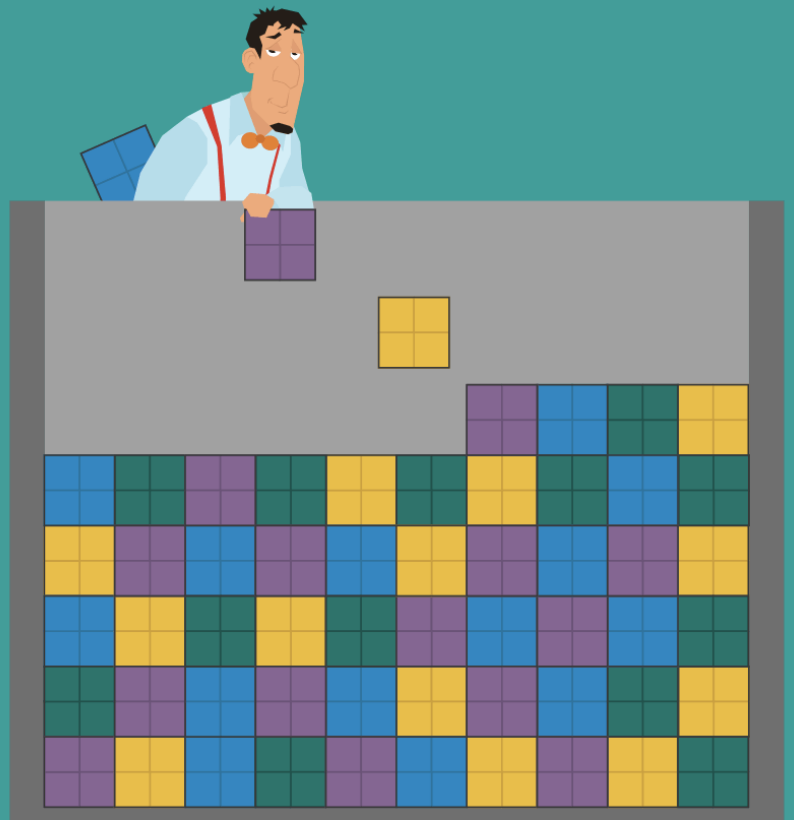
Build snowflakes for each environment



AFTER

Build once, run anywhere

A shared construct between software and hardware



DaoCloud

IT大咖说 知识分享平台

建立持续交付流水线

为持续交付做准备

DaoCloud

关于持续交付

- 持续交付是一组能够帮助软件开发团队极大的提高其软件交付的速度和质量的模式和最佳实践组成。
- 优势：
  - **快**： 尽可能快地交付软件，尽可能早地将有价值的新功能运用于生产
  - **好**：提高软件质量，系统正常运行时间和稳定性
  - **好**： 降低发布风险，避免同时在测试和生产环境部署失败
  - **好**： 减少浪费，提高开发和交付过程的效率
  - **快**：使您的软件始终处于生产就绪状态，以便您可以随时部署

预备步骤

- 自动化测试等开发实践
- 软件结构和组件设计，可帮助做更频繁的发布，而不影响用户，包括功能标志
- 工具如源代码管理，持续集成，配置管理和应用发布自动化软件
- 自动化和脚本化，使您能够以有限的人为干预重复构建，打包，测试，部署和监控软件
- 组织，文化和业务流程的变化，以支持持续交付

核心：自动化

持续交付第一步：**自动化**

- 自动化构建和打包
- 自动化持续集成
- 自动化测试
- 自动化部署
- 受管理的基础架构和云
- 基础架构即代码
- 容器框架
- 自动化生产部署

自动化测试

TEST TYPE	TO CONFIRM THAT
Unit Tests	Low level functions and classes work as expected under a variety of inputs.
Integration Tests	Integrated modules work together and in conjunction with infrastructure such as message queues and databases.
Acceptance Tests	Key user flows work when driven via the user interface, regarding your application as a complete black box.
Load Tests	Your application performs well under simulated real world user load.
Performance Tests	The application meets performance requirements and response times under real world load scenarios.
Simulation Tests	Your application works in device simulation environments. This is especially important in the mobile world where you need to test software on diverse emulated mobile devices.
Smoke Tests	Tests to validate the state and integrity of a freshly deployed environment.
Quality Tests	Application code is high quality – identified through techniques such as static analysis, conformance to style guides, code coverage etc.

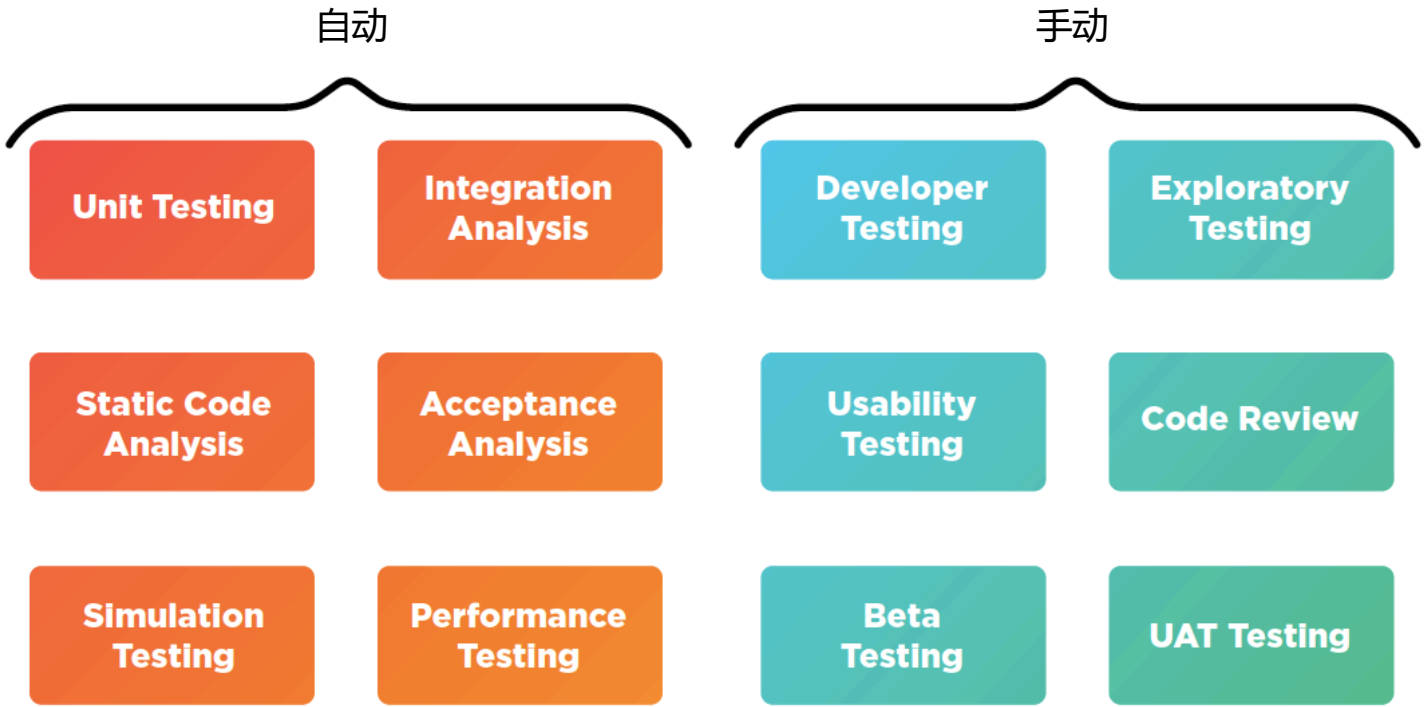
持续交付第二步： 实现一个持续交付流水线

- 流水线建模
- 识别非自动化的活动和网关
- 实现流水线

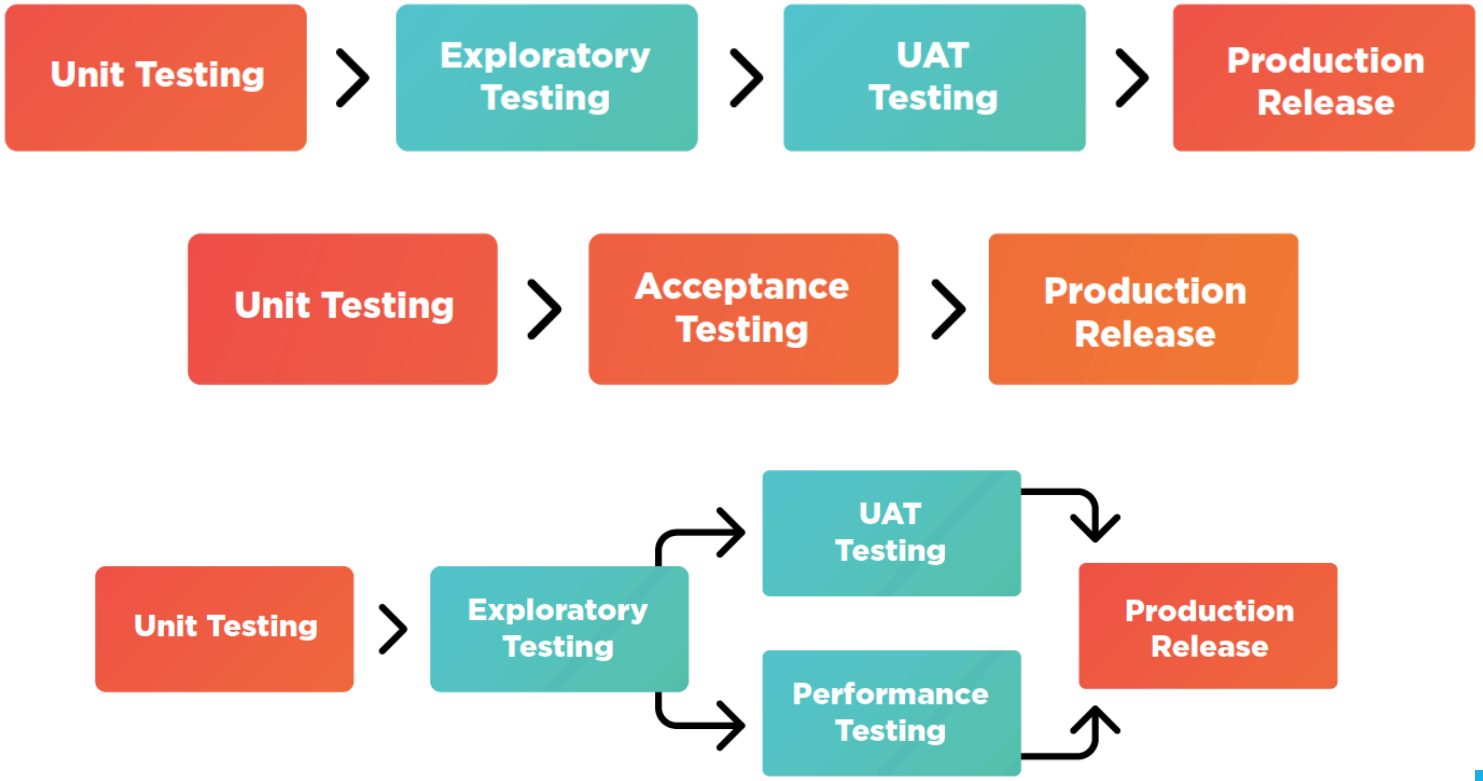
什么是流水线

- 软件在从源代码到生产上线之间的各显式阶段
- 哪些阶段是自动化的以及哪些阶段包含手工步骤
- 在流水线各阶段之间推动的标准是什么，捕获哪些网关是自动化的，哪些是手动的
- 哪些步骤可以并行

流水线建模（一）



流水线建模（二）


Unit Testing
Exploratory Testing
UAT Testing
Production Release
Unit Testing
Acceptance Testing
Production Release
Unit Testing
Exploratory Testing
UAT Testing
Performance Testing
Production Release

流水线建模（三）

IDEAL SITUATION	TRADEOFF
All stages and gateways would be automated.	Requires substantial investment in automated tests and release automation.
Always avoid expensive human re-work.	You need to parallelise test phases if they are slow and manual, giving the risk of failed release candidates in another stage of the pipeline.
Always perform automated testing.	Detailed automated testing is also expensive in production like environments.
Implement lots of environments to support testing phases.	Maintaining environments has an associated management and financial cost.

持续交付第三步： 最佳实践

- 实现监测
- 实现回滚
- 提取特定于环境的配置（配置与代码分离）
- 执行金丝雀发布
- 记录审计信息
- 实现功能开关
- 使用基于云的基础架构

DaoCloud

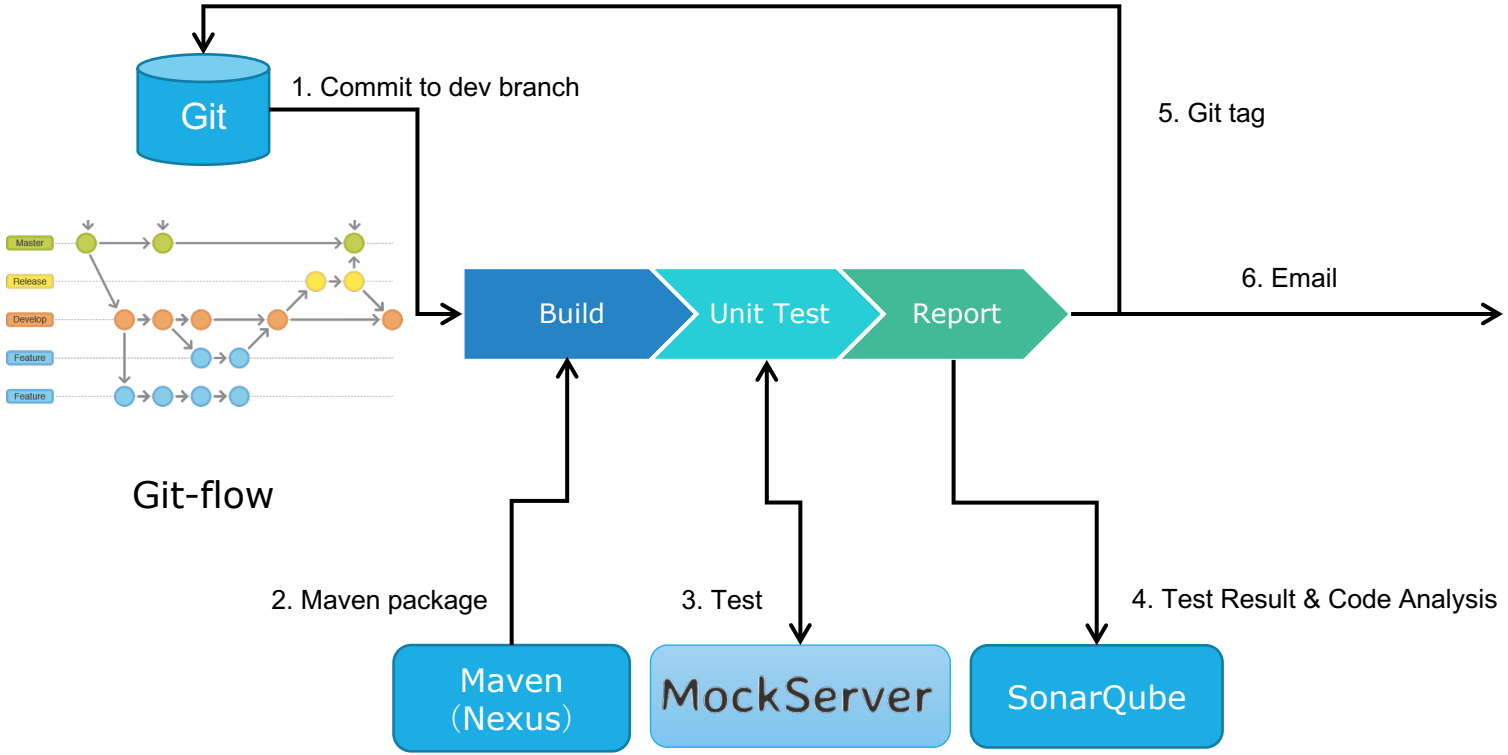
IT大咖说 知识分享平台

基于Spring Cloud微服务应用的持续交付最佳实践

持续集成 / 持续交付

DaoCloud

持续集成


Git
1. Commit to dev branch
5. Git tag
Master
Release
Develop
Feature
Feature
Build
Unit Test
Report
6. Email
Git-flow
2. Maven package
3. Test
4. Test Result & Code Analysis
Maven
(Nexus)
MockServer
SonarQube

持续部署

8. Git tag

环境A

DaoCloud

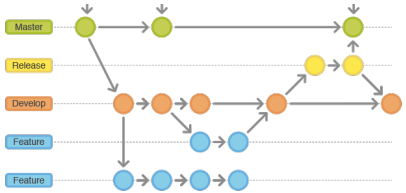
IT大咖说 知识分享平台

微服务

A C D F

Git

1. Merge dev branch to uat branch



Git-flow

6. Deploy

7. Test

TC

(Maven) Build

(Maven) Deploy

(Docker) Build/Push

(Docker) Deploy

Integration Test

3. Maven deploy

4. Docker build

2. Maven package

5. Docker push

Maven (Nexus)

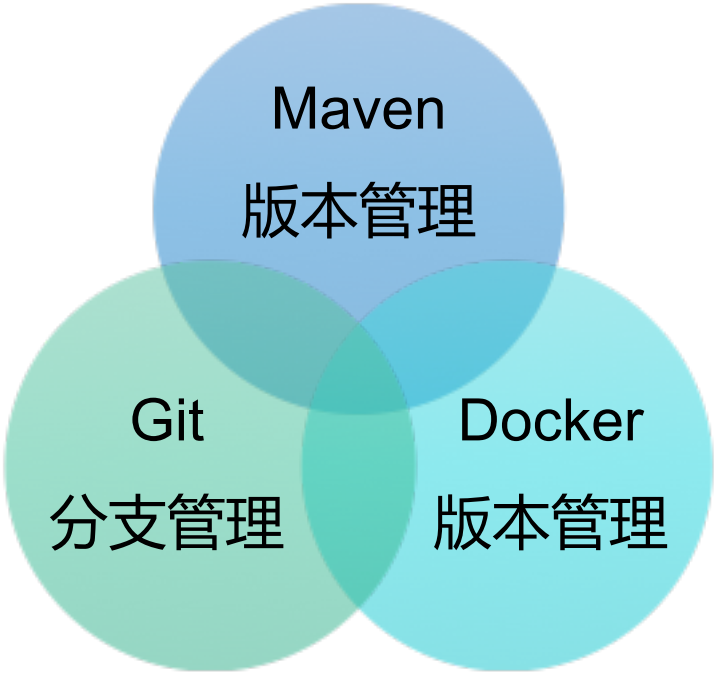
Docker Registry

环境B

微服务

A C D F

挑战一：版本管理



- Maven版本
  - 1.0.0
  - Snapshot
  - Release
- Git分支
  - Dev
  - Uat
  - Release
- Docker镜像版本（tag）
  - Latest
  - v1.0

挑战一：思考

- version外部管理（可以提供API进行获取），例如
  - /{service.name}/versions/latest
  - /{service.name}/versions
- Maven设置版本，与Git分支对应
  - mvn versions:set -DnewVersion=1.0.0-{git分支名}
  - 测试稳定之后，git打同名tag
- Docker镜像的版本和Maven的版本一致

挑战二：部署

- 服务编排
  - 一组相关联微服务如何来描述？
- 服务部署
  - 一组微服务如何依次部署？
- 环境管理
  - 多套环境如何管理？

挑战二：思考

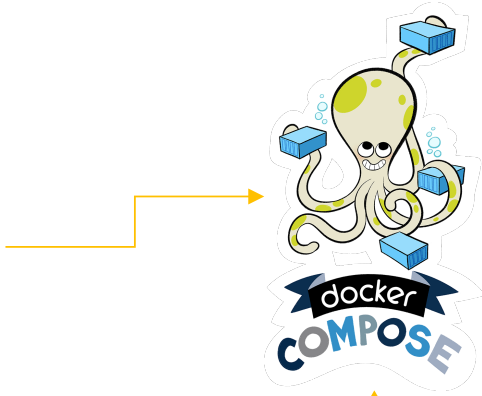
1. Compose YAML

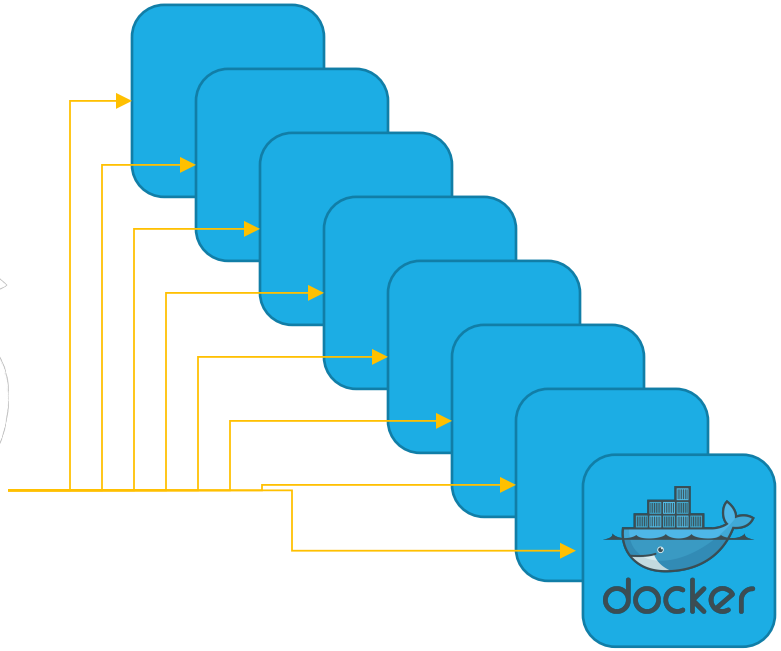
```
version: "2"

services:
    tyk_dashboard:
        image: daocloud.io/daocloud/tyk-dashboard
        environment:
            - LICENSE_KEY={{ LICENSE_KEY }}
            - OVERRIDE_HOSTNAME=tyk.co
            - NODE_IS_SEGMENTED=false
            - MONGO_USER={{ MONGO_USER }}
            - MONGO_PASS={{ MONGO_PASS }}
            - MONGO_DB={{ MONGO_DB }}
            - MONGO_HOST={{ MONGO_HOST }}
            - REDIS_HOST={{ REDIS_HOST }}
            - REDIS_PORT={{ REDIS_PORT }}
        ports:
            - 3000
            - 5000

    tyk_pump:
        image: daocloud.io/daocloud/tyk-pump
        environment:
            - MONGO_USER={{ MONGO_USER }}
            - MONGO_PASS={{ MONGO_PASS }}
            - MONGO_DB={{ MONGO_DB }}
            - MONGO_HOST={{ MONGO_HOST }}
            - REDIS_HOST={{ REDIS_HOST }}
            - REDIS_PORT={{ REDIS_PORT }}

    tyk_gateway:
        image: daocloud.io/daocloud/tyk-gateway
        ports:
            - 8080
        environment:
            - TYK_GW_ALLOWINSECURECONFIGS=true
            - MONGO_USER={{ MONGO_USER }}
            - MONGO_PASS={{ MONGO_PASS }}
            - MONGO_HOST={{ MONGO_HOST }}
            - MONGO_DB={{ MONGO_DB }}
            - REDIS_HOST={{ REDIS_HOST }}
            - REDIS_PORT={{ REDIS_PORT }}
```





DaoCloud IT大咖说 知识分享平台

配置中心

DaoCloud

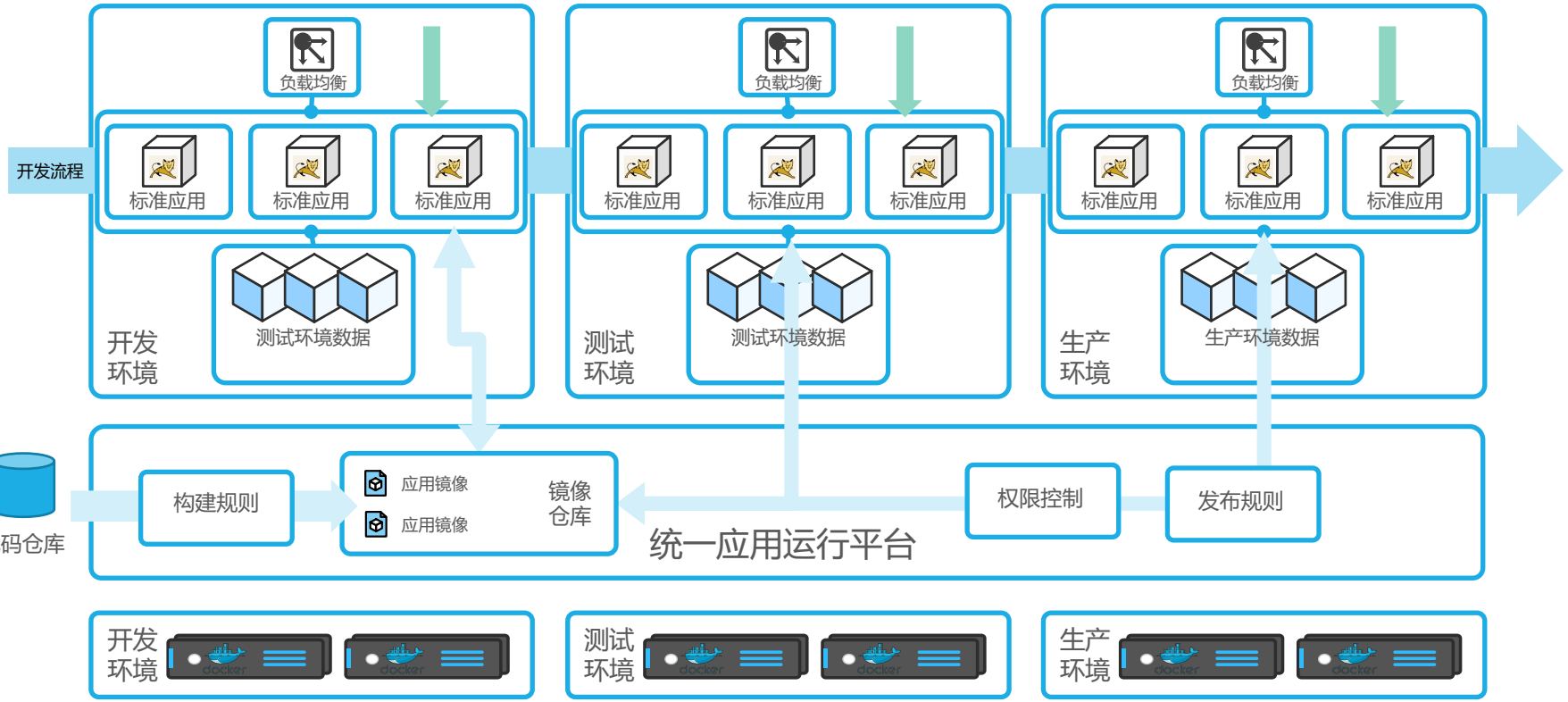
DaoCloud

IT大咖说 知识分享平台

在DaoCloud上的实践

DaocCloud Services & DaoCloud Enterprise

建设目标：应用自动化交付



Q&A