企业级持续交付平台的微服务演进

[ThoughtWorks 林帆]

- 目录 -

- 一个案例：持续交付平台
- 你以为微服务就是拆服务？
- 微服务的容器化探索

林帆
ThoughtWorks DevOps技术咨询师
《CoreOS实践之路》书作者

案例：企业级持续交付平台

- 2015年8月 -

- ~5人临时团队
- 1个Service (和其他开源服务)
- 不到10服务节点
- 接入项目数~50

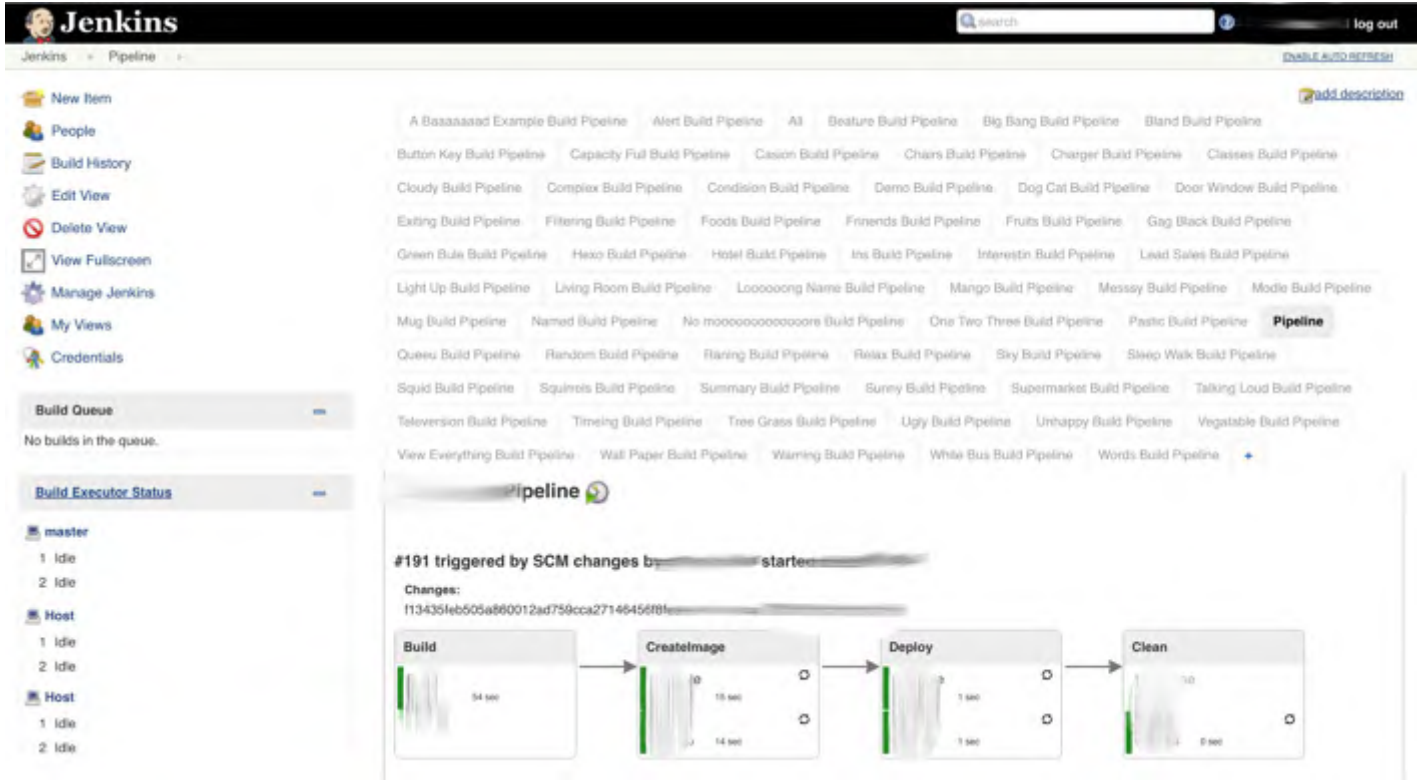
- 2016年8月 -

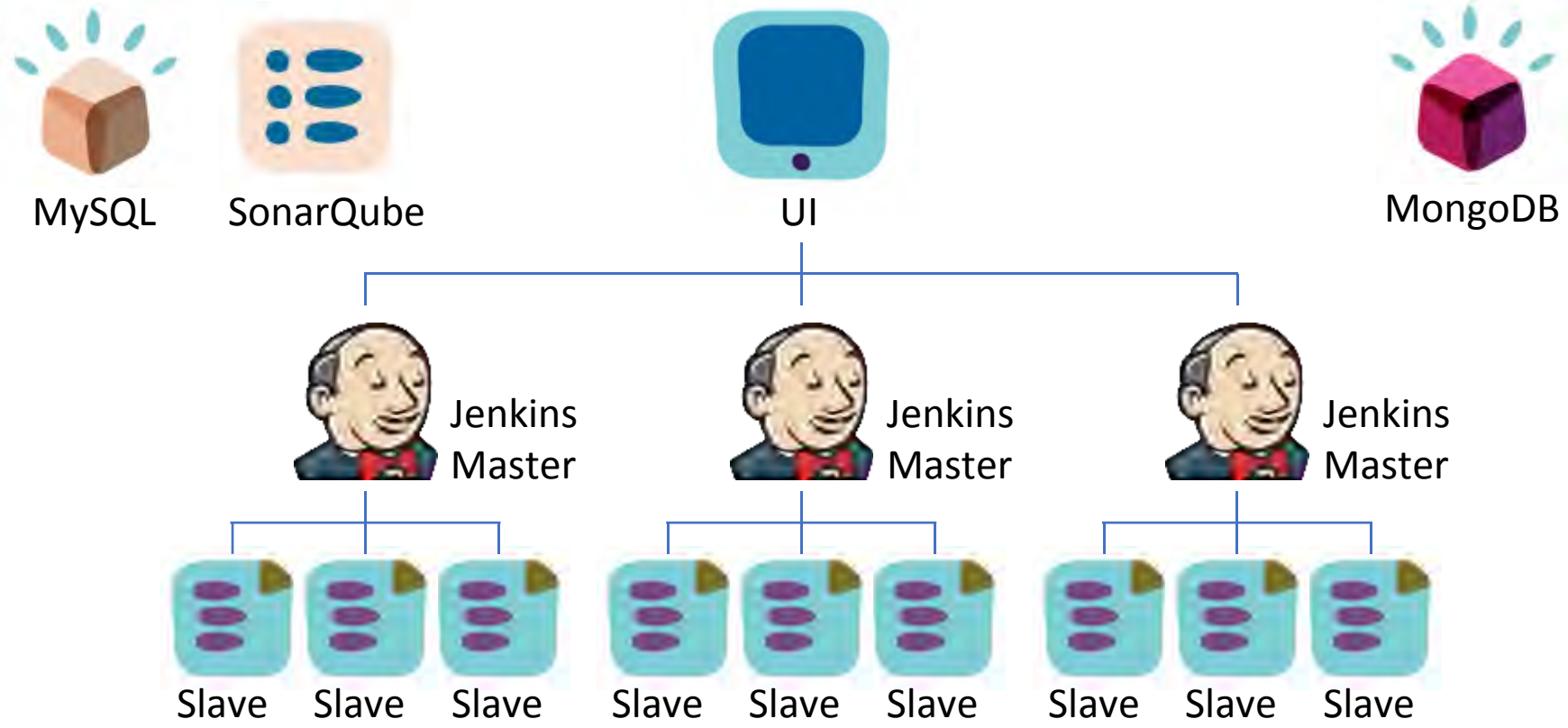
- ~20人，分为两个团队
- 超过20个Service
- 超过200服务节点
- 接入项目数超过1000

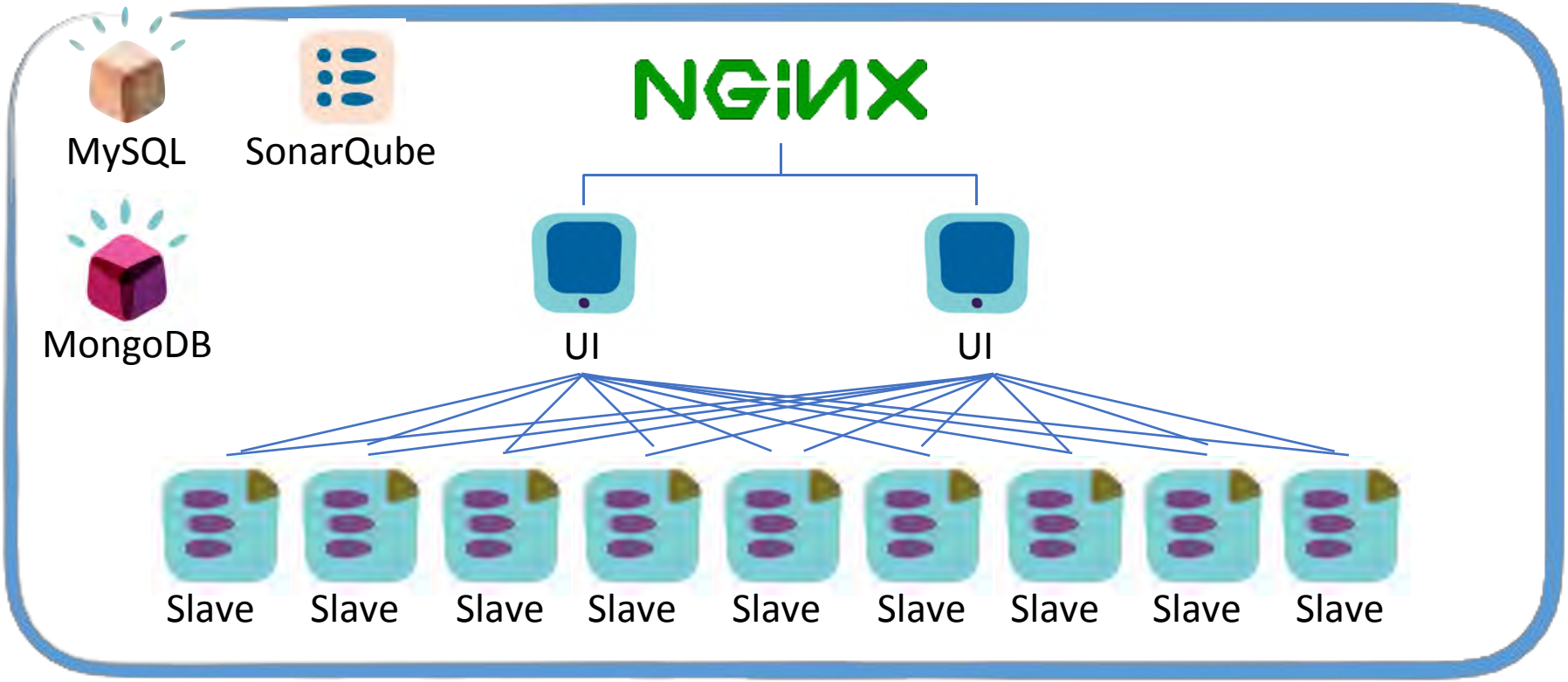
Jenkins

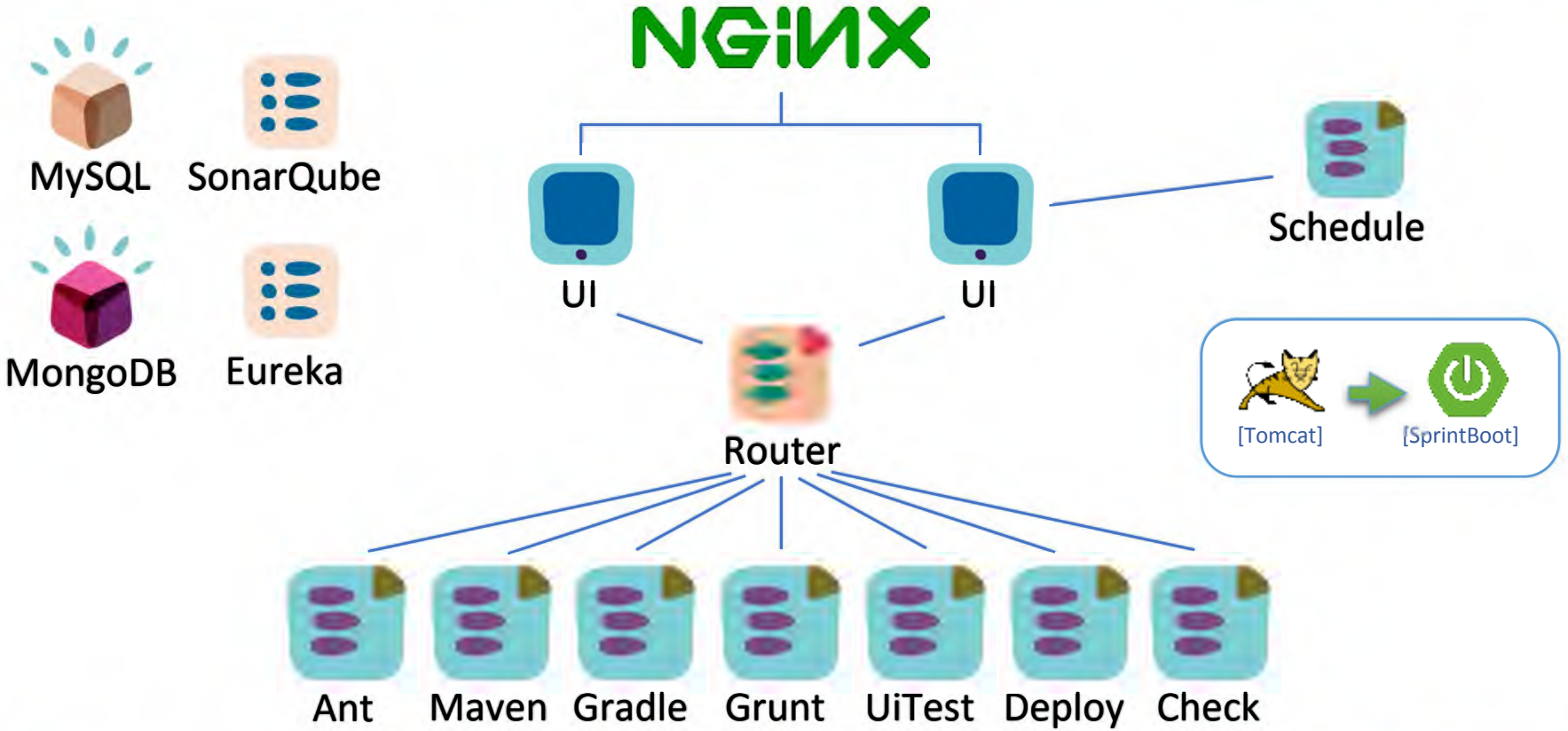
Atlassian Bamboo

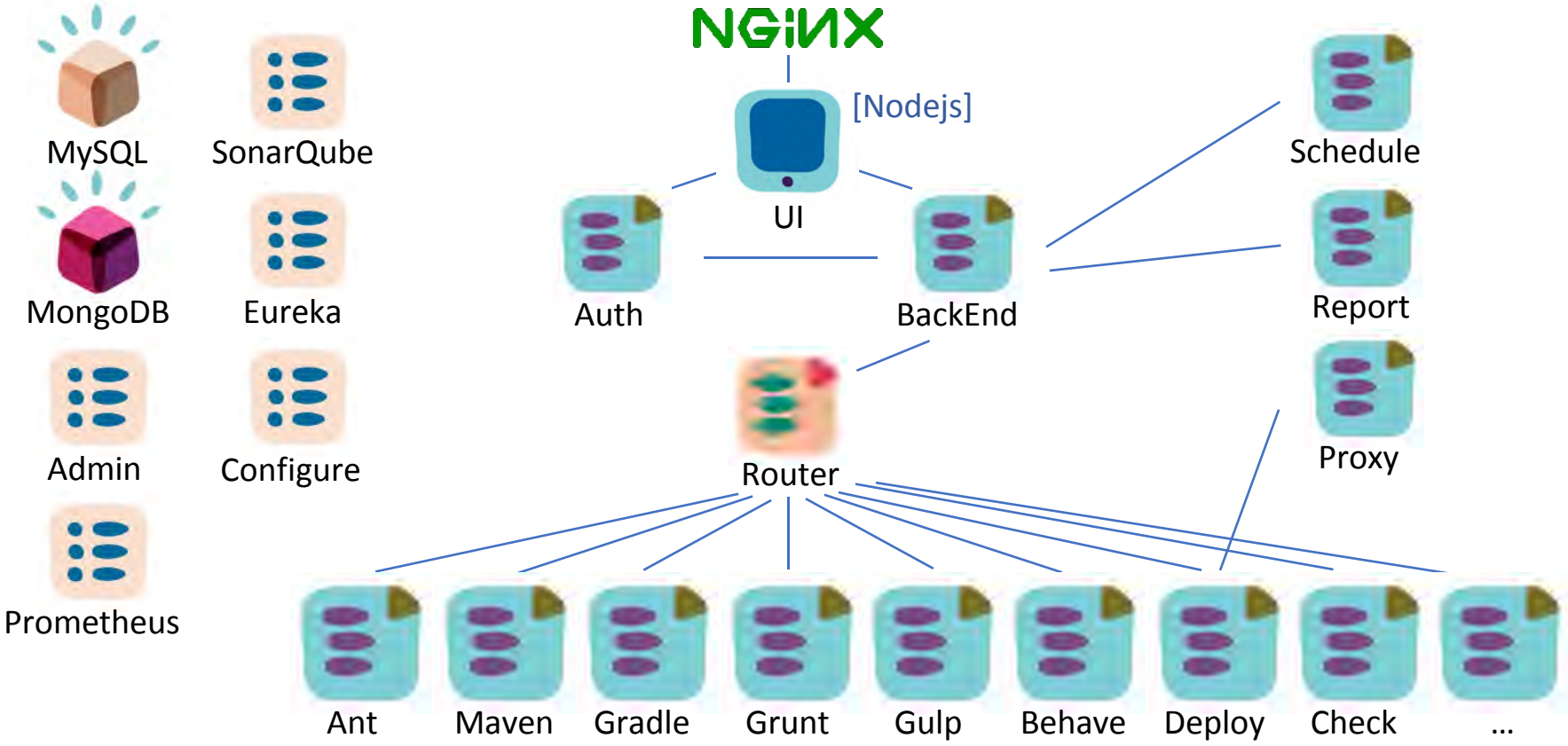
go

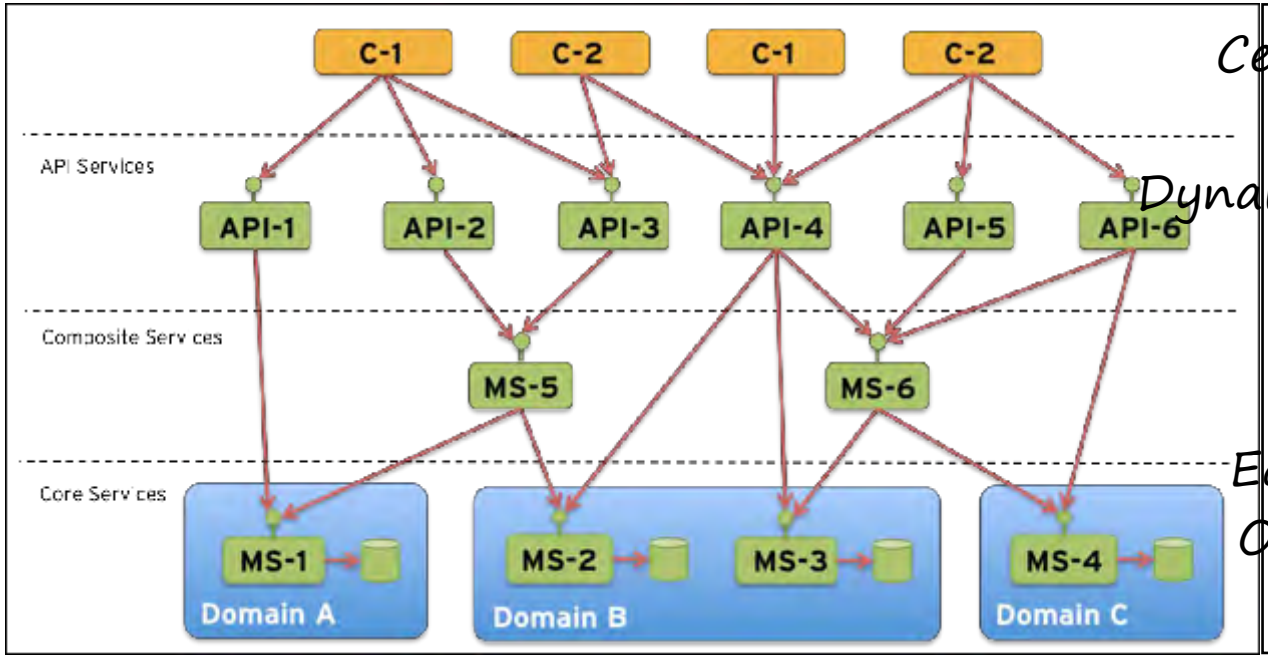

Jenkins
log out
Jenkins » Pipeline »
ENABLE AUTO REFRESH
New Item
People
Build History
Edit View
Delete View
View Fullscreen
Manage Jenkins
My Views
Credentials
add description
Pipeline
Build Queue
No builds in the queue.
Build Executor Status
master
1 Idle
2 Idle
Host
1 Idle
2 Idle
Host
1 Idle
2 Idle
#191 triggered by SCM changes b
Changes:
Build
CreateImage
Deploy
Clean


MySQL
SonarQube
UI
MongoDB
Jenkins Master
Jenkins Master
Jenkins Master
Slave
Slave
Slave
Slave
Slave
Slave
Slave
Slave
Slave


MySQL
SonarQube
NGINX
MongoDB
UI
UI
Slave
Slave
Slave
Slave
Slave
Slave
Slave
Slave
Slave

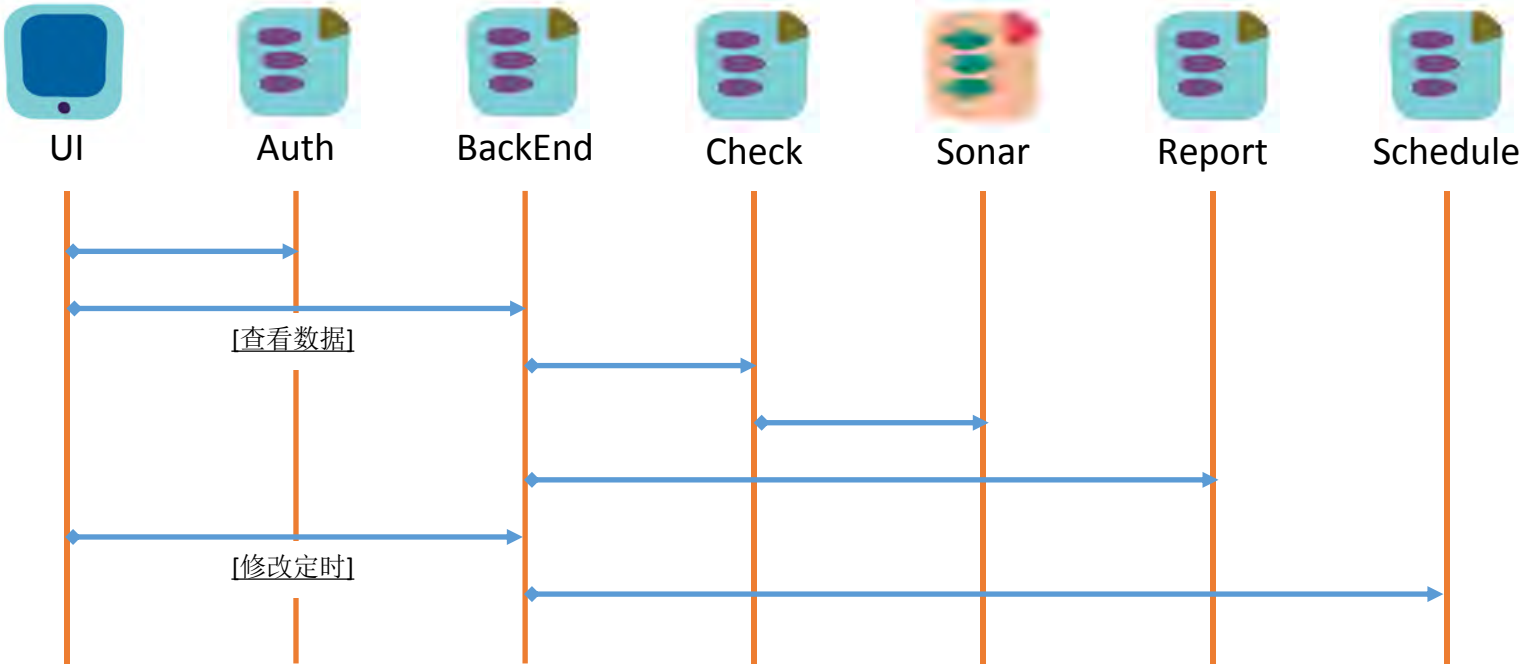

NGINX
MySQL
SonarQube
MongoDB
Eureka
UI
UI
Schedule
Router
[Tomcat]
[SprintBoot]
Ant
Maven
Gradle
Grunt
UiTest
Deploy
Check


NGINX
[Nodejs]
UI
Auth
BackEnd
Router
Schedule
Report
Proxy
MySQL
SonarQube
MongoDB
Eureka
Admin
Configure
Prometheus
Ant
Maven
Gradle
Grunt
Gulp
Behave
Deploy
Check
...

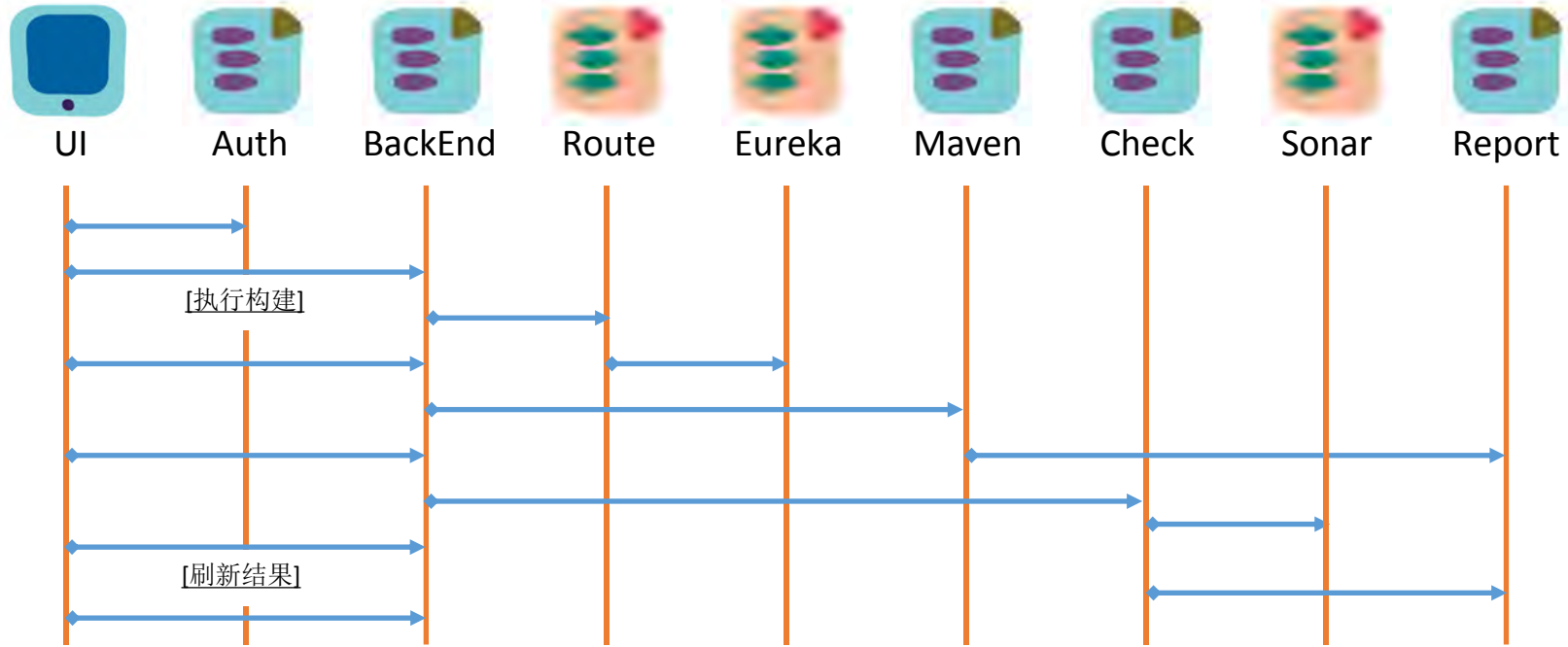


Central Configuration server
Service Discovery server
Dynamic Routing & Load Balan
Circuit Breaker
Monitoring
Centralised log analysis
Edge Server (API Gateway)
OAuth 2.0 protected API's

用户查看和修改流水线



用户执行一次项目构建


UI
Auth
BackEnd
Route
Eureka
Maven
Check
Sonar
Report
[执行构建]
[刷新结果]

服务何时应该拆分？

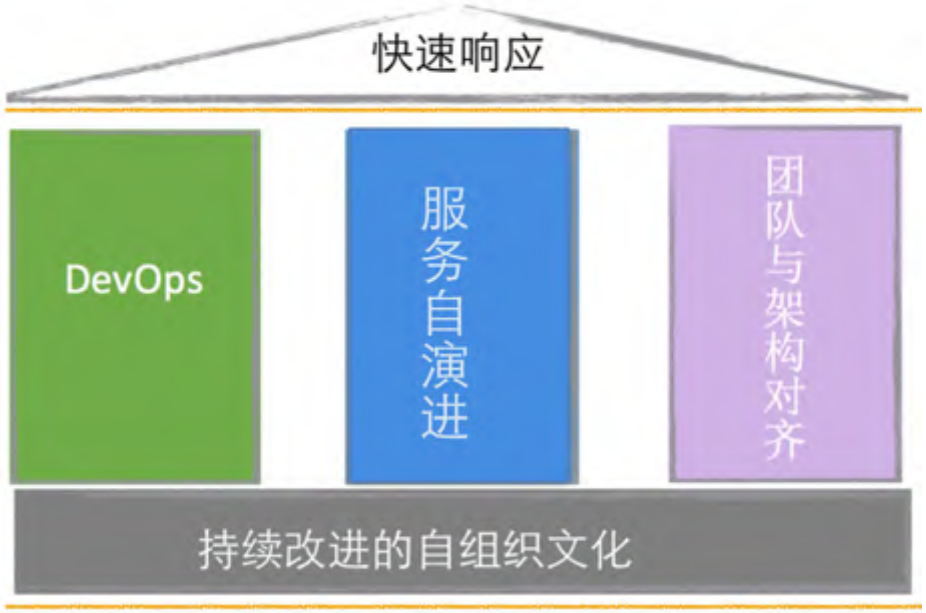
- 代码量过大、逻辑混乱（也许可以考虑重写这个服务...）
- 服务承担了多个独立业务职责（代码存在大量重复判断语句）
- 同时有太多功能需要在同一个代码仓库进行开发（>3~5个）

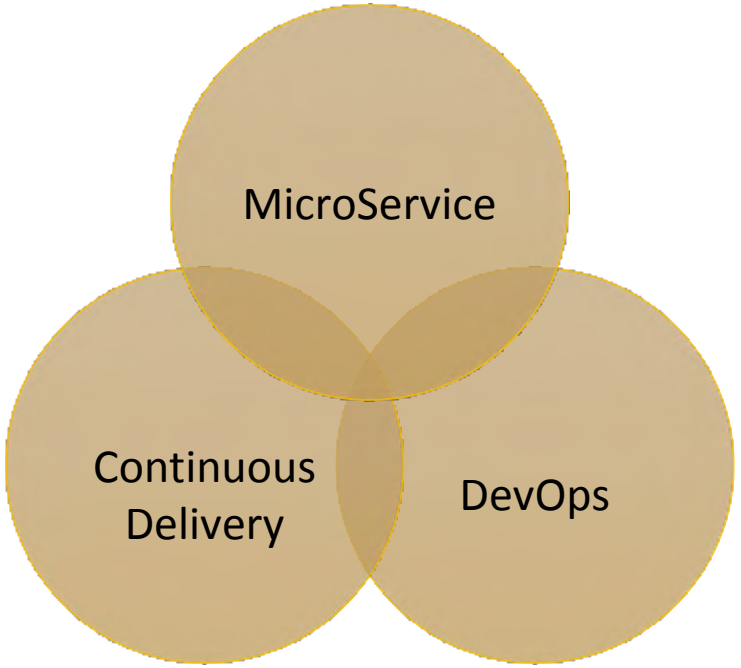
你以为微服务的关键真的就是拆分服务？

微服务真的只是拆分服务？

- 架构不是银弹
- 基础设施&团队文化

设计系统的组织，其产生的设计和架构等价于组织间的沟通结构
—— Conway’ s law

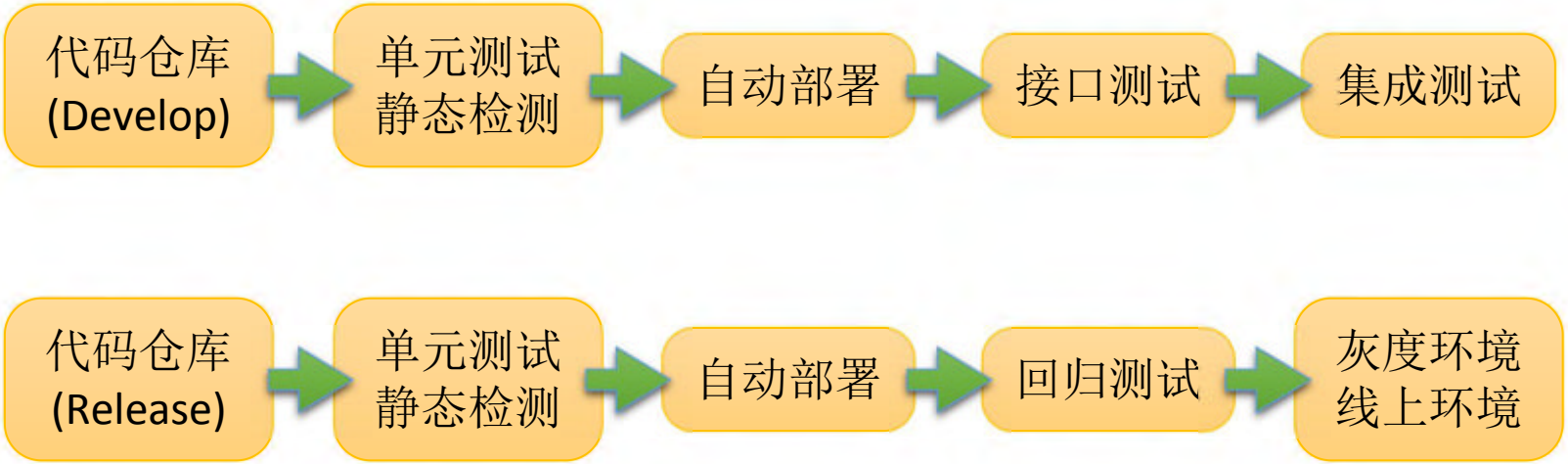

快速响应
DevOps
服务自演进
团队与架构对齐
持续改进的自组织文化


MicroService
Continuous Delivery
DevOps

微服务关键实践

- 持续交付
- 全功能团队
- 自动化运维
- 服务高可用
- 不离线部署
- 监控告警
- 容器化

基于主干开发的流水线

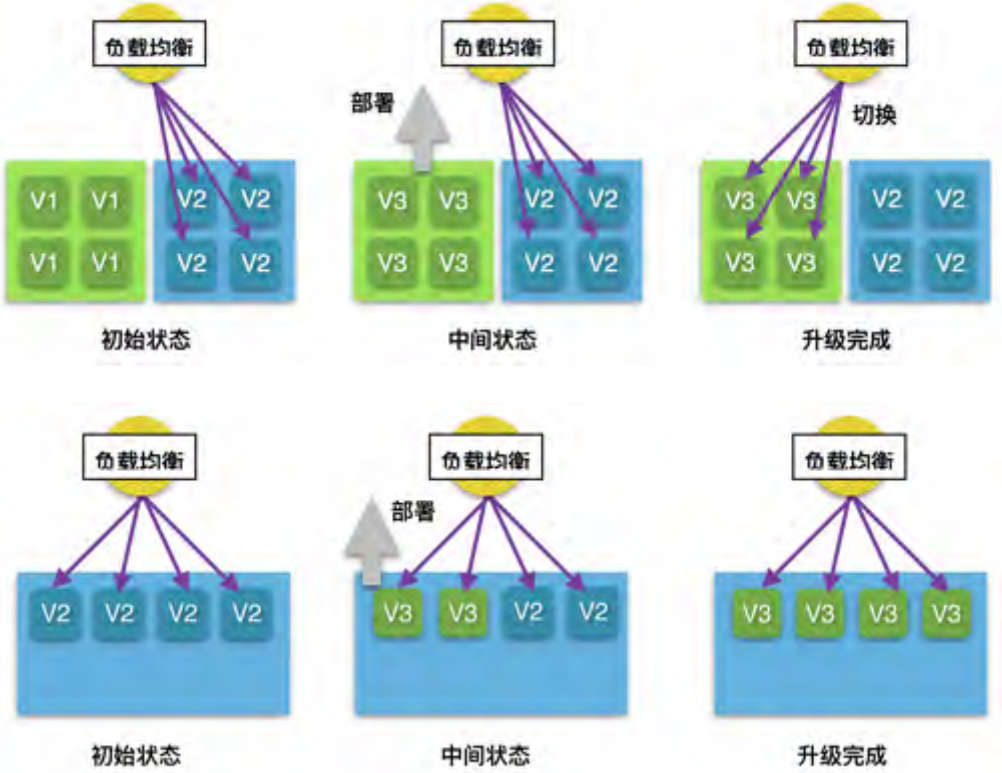

代码仓库
(Develop)
单元测试
静态检测
自动部署
接口测试
集成测试
代码仓库
(Release)
单元测试
静态检测
自动部署
回归测试
灰度环境
线上环境

服务的不离线部署

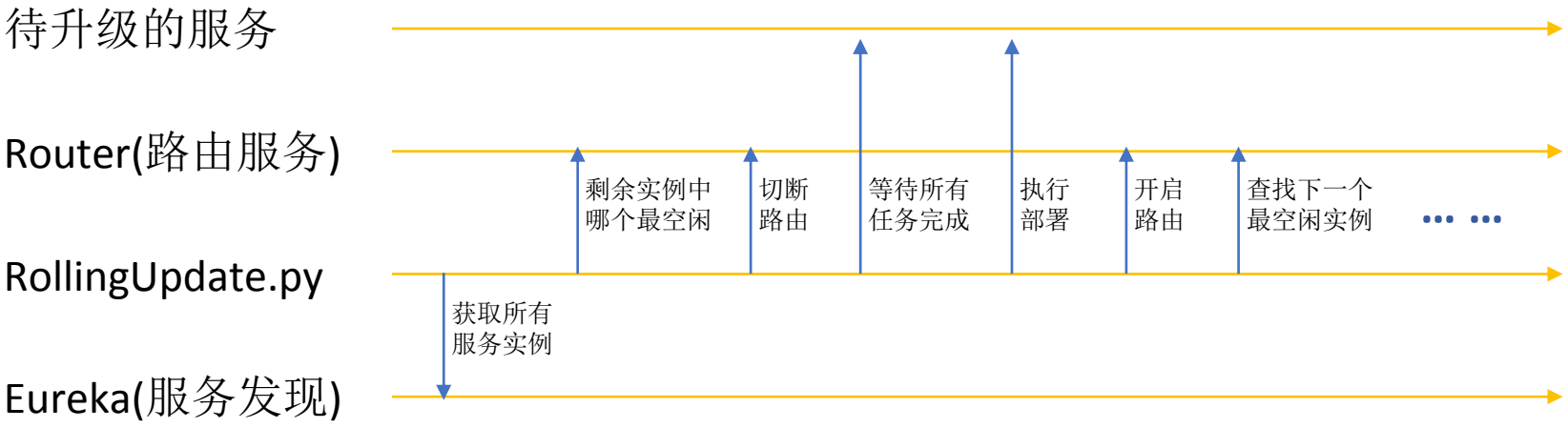
BlueGreen Update

V.S.

Rolling Update



带状态检查的Rolling Update



为什么要用微服务

- 多技术栈混用，方便新技术引入
- 更低的试错成本，更快的响应变化

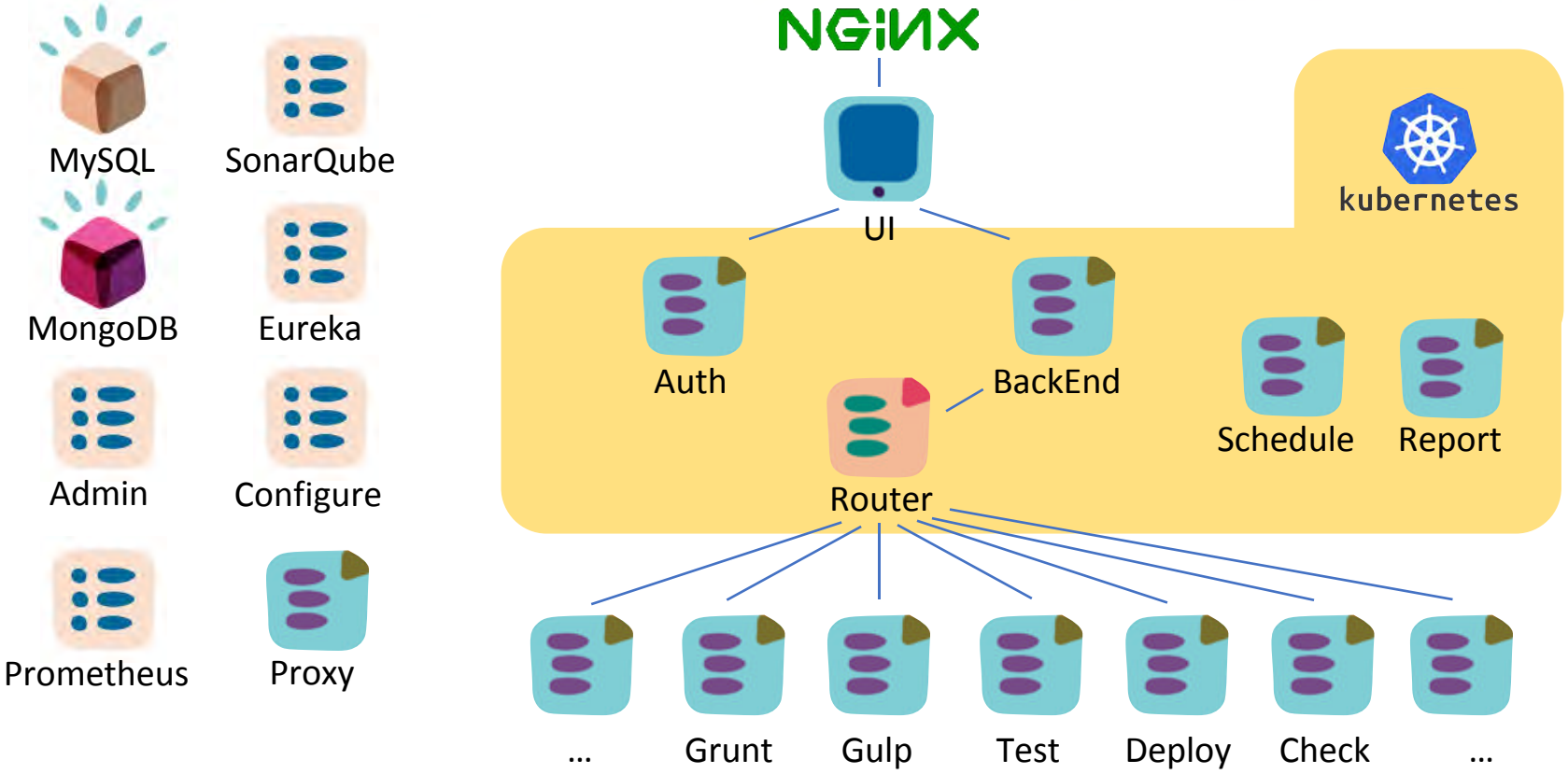
微服务的潜在好处

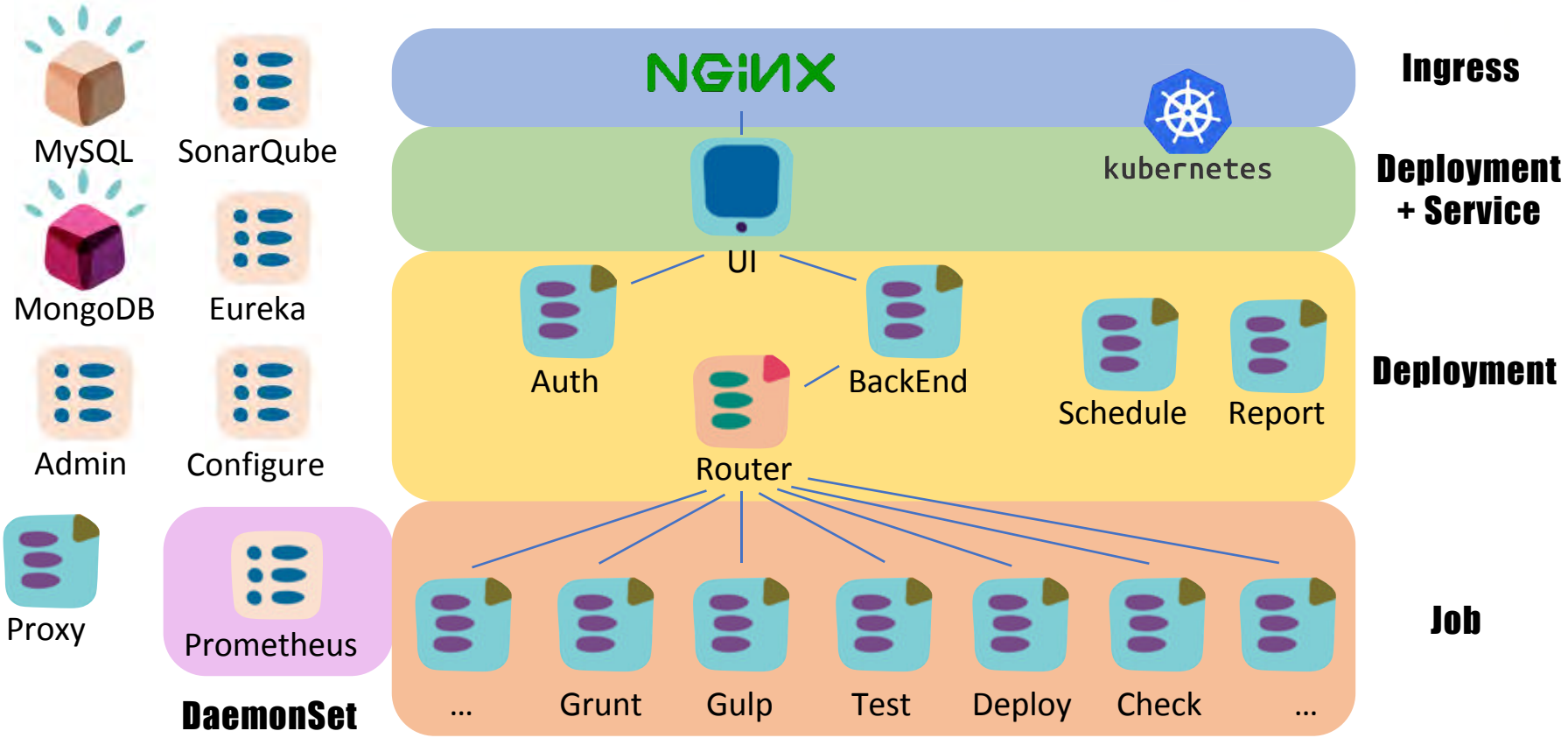
- 更清晰的代码，更少的分支，便于持续集成优化
- 更短的构建时间，更快的运行测试，更好的并行开发

微服务的潜在问题

- 接口&依赖的版本管理
- 链式故障&跨团队问题追踪
- 数据表归属&数据耦合
- 分布式系统的其他一切问题

微服务的容器化探索


MySQL
SonarQube
MongoDB
Eureka
Admin
Configure
Prometheus
Proxy
NGINX
UI
kubernetes
Auth
BackEnd
Schedule
Report
Router
...
Grunt
Gulp
Test
Deploy
Check
...


MySQL
SonarQube
MongoDB
Eureka
Admin
Configure
Proxy
Prometheus
DaemonSet
NGINX
kubernetes
Ingress
UI
Deployment
+ Service
Auth
BackEnd
Router
Schedule
Report
Deployment
...
Grunt
Gulp
Test
Deploy
Check
...
Job

Thanks

flin@thoughtworks.com