

风控系统在容器化时代的实践

朱清

冰鉴科技技术总监

QCon

全球软件开发大会

10月17-19日 上海·宝华万豪酒店



扫码锁定席位

九折即将结束

团购还享更多优惠，折扣有效期至9月17日

扫描右方二维码即可查看大会信息及购票



如果在使用过程中遇到任何问题，可联系大会主办方，欢迎咨询！

微信：qcon-0410

电话：010-84782011

ArchSummit

全球架构师峰会 2017



扫码锁定席位

12月8-9日 北京·国际会议中心

七折即将截止立省2040元

使用限时优惠码AS200，

以目前最优惠价格报名ArchSummit

仅限前20名用户，优惠码有效期至9月19日，

扫描右方二维码即可使用



如果在使用过程中遇到任何问题，可联系大会主办方，欢迎咨询！

微信：aschina666

电话：15201647919

极客搜索

全站干货，一键触达，只为技术

s.geekbang.org



扫描二维码立即体验

有没有一种搜索方式，能整合 InfoQ 中文站、极客邦科技旗下12大微信公众号矩阵的全部资源？

极客搜索，这款针对极客邦科技全站内容资源的轻量级搜索引擎，做到了！

扫描上方二维码，极客搜索！

这里只有 技术领导者

EGO会员第二季招募季正式开启



E小欧

报名时间：9月1日-9月15日

扫描添加E小欧，
邀您进入EGO会员预报名群

立即报名



TABLE OF CONTENTS

向着微服务进军

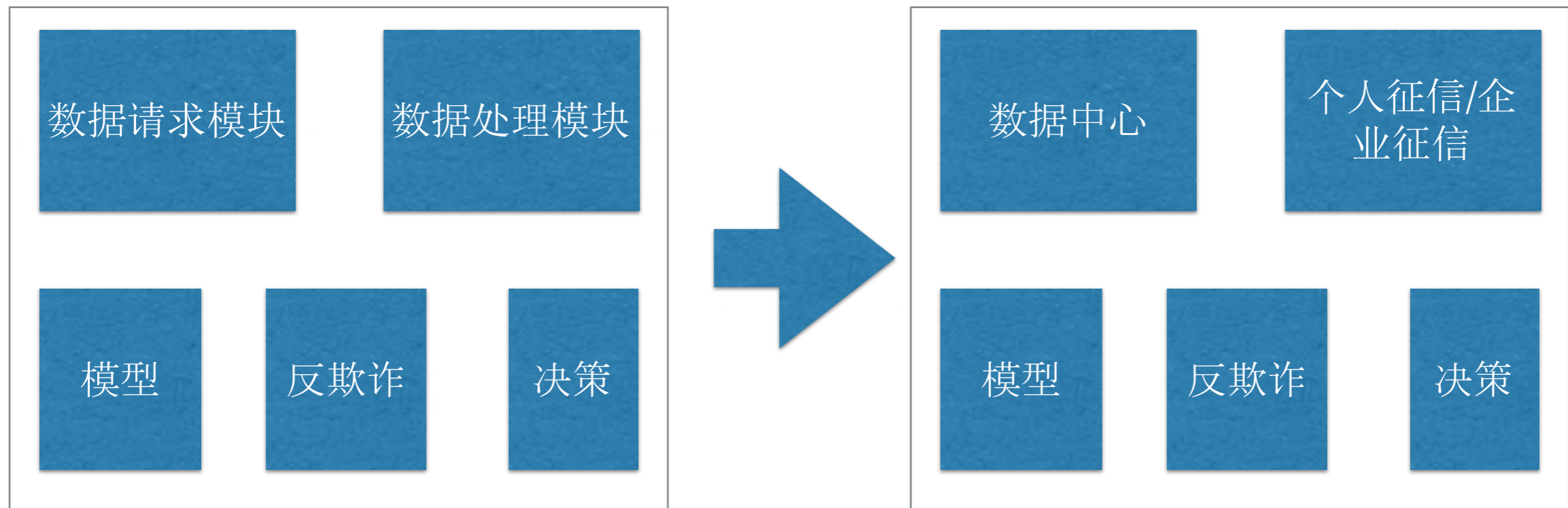
演进中的问题

基于容器技术的解决方案

风控平台核心流程



换代前



换代后的微服务架构

- 去年底，在经过了几个月的研究后，决定上微服务架构。

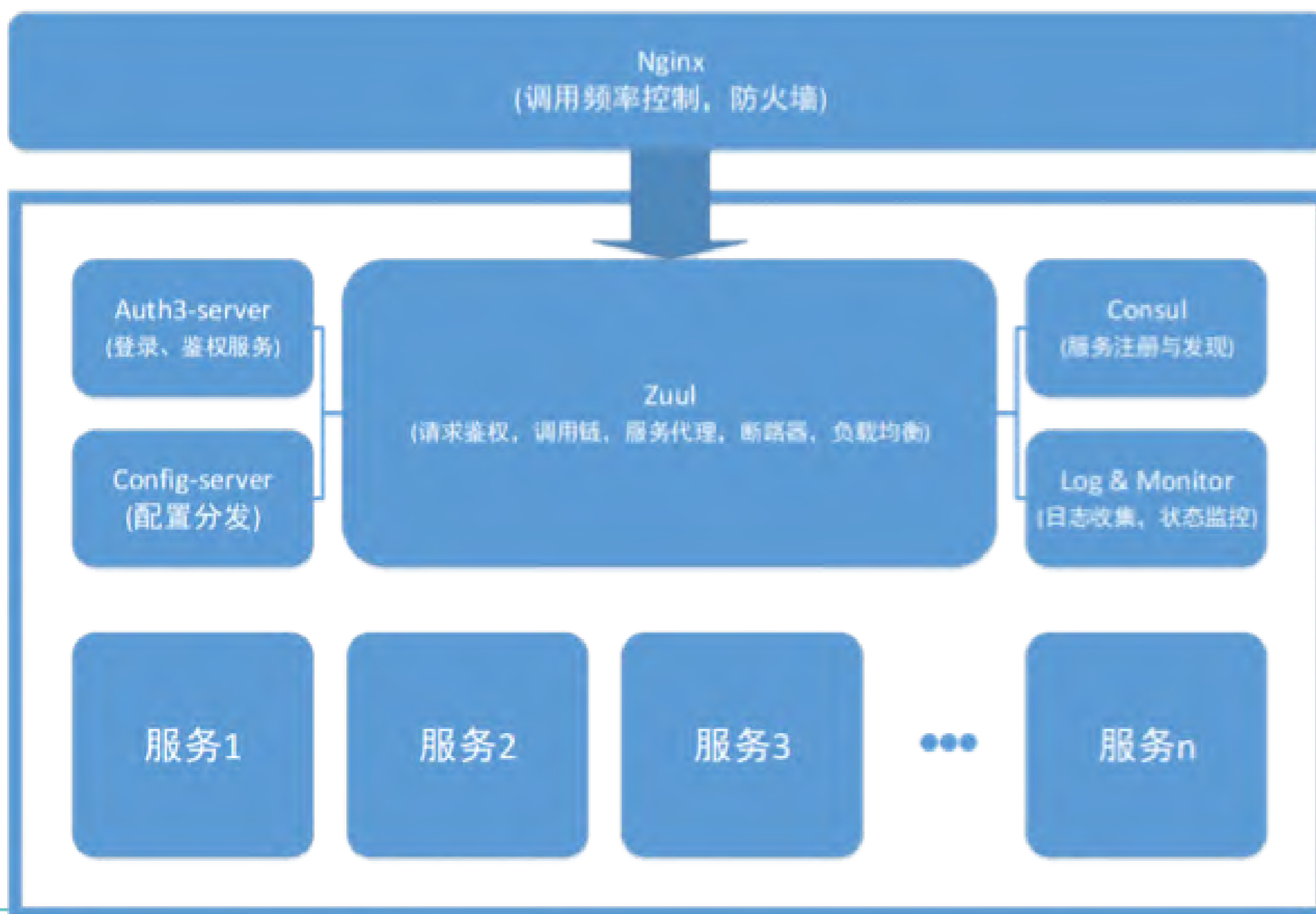


TABLE OF CONTENTS

向着微服务进军

演进中的问题

基于容器技术的解决方案

演进中的问题

- 快速发布：服务拆分后带来的部署、管理问题

相比于一个整合了数个独立模块的单体应用，服务拆分之后，虽然可以很快将一个微小的改动快速更新而不用针对整个应用更新。但拆分后运维的部署工作无疑随着独立服务的数量增加而变得多而杂，对于我们只有2个人的运维团队，必然非常的吃力。

演进中的问题

- 成本考虑：多样化的需求、服务数量激增

不同业务性能的要求差异比较大，服务拆分后，服务数量飙升，为了做到资源、环境隔离，按原有的方式部署机器利用率有限，成本很高。

演进中的问题

- 配置分散：不同环境不同服务的配置数量激增

同样地，对于应用所使用到的一些配置信息，在单体应用时期，不同的环境做一下文件替换就能解决，但是当服务拆分后体量上增加一个数量级时，这也会带来巨大的问题，并且分散后管理上也将带来很大的困难。

演进中的问题

- 日志留存、快速定位：业务激增、流程复杂带来的存储问题

一次征信服务的流程比较复杂，涉及授权、外部数据调用、网络数据采集、数据处理、模型/反欺诈、策略的调用，服务分散后，问题定位在日志中显得非常混乱，此外随着业务量的极速增长，分散到对各个服务后采集和存储也面临着新的挑战。

TABLE OF CONTENTS

向着微服务进军

演进中的问题

基于容器技术的解决方案

TABLE OF CONTENTS

向着微服务进军

演进中的问题

基于容器技术的解决方案

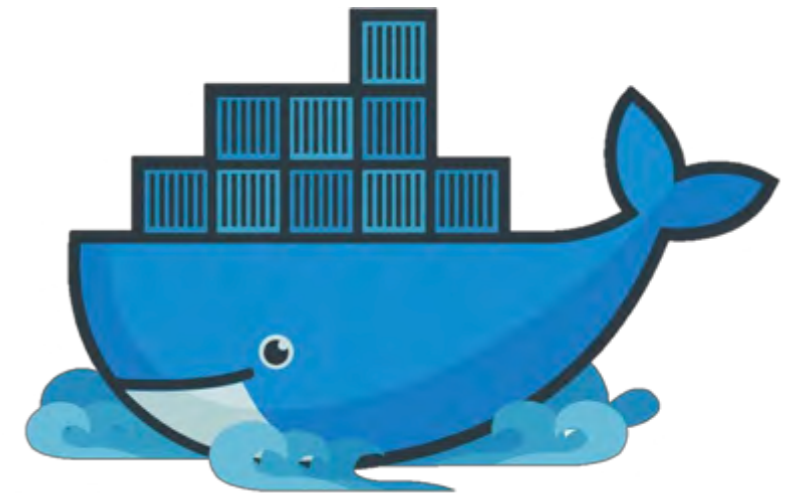
容器引擎

快速发布实践

与服务注册的结合

容器监控

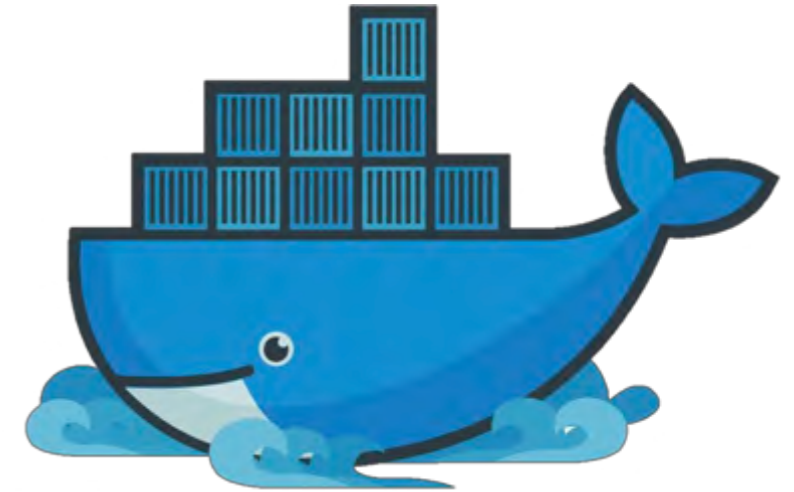
容器引擎



- 快速发布的基础

“服务作为不可变设施”这种模式有很多好处，配置工作的可重复性，减少了配置管理的工作，让后续的持续集成与持续部署过程变得更流畅。同时它也更易于应对部署环境间的差异及版本管理，包括在部署出错时可进行快速回滚。同时，它能让你将运行环境和配置放在代码中，然后启动一个镜像的实例就可以快速部署一个应用

容器引擎



- 成本控制的基础

Docker只是创建一个容器进程而无需启动操作系统，这个过程只需要秒级的时间，我们可以快速根据服务的运行能力进行扩容

由于没有多个操作系统的内存占用，以及能在多个实例之间共享没有使用的内存，**Docker**的虚拟化可以比虚拟机提供更好的服务器整合解决方案，这种虚拟化带来的损耗比较低

使用**Docker**并进行有效的资源分配可以提高资源的利用率

TABLE OF CONTENTS

向着微服务进军

演进中的问题

基于容器技术的解决方案

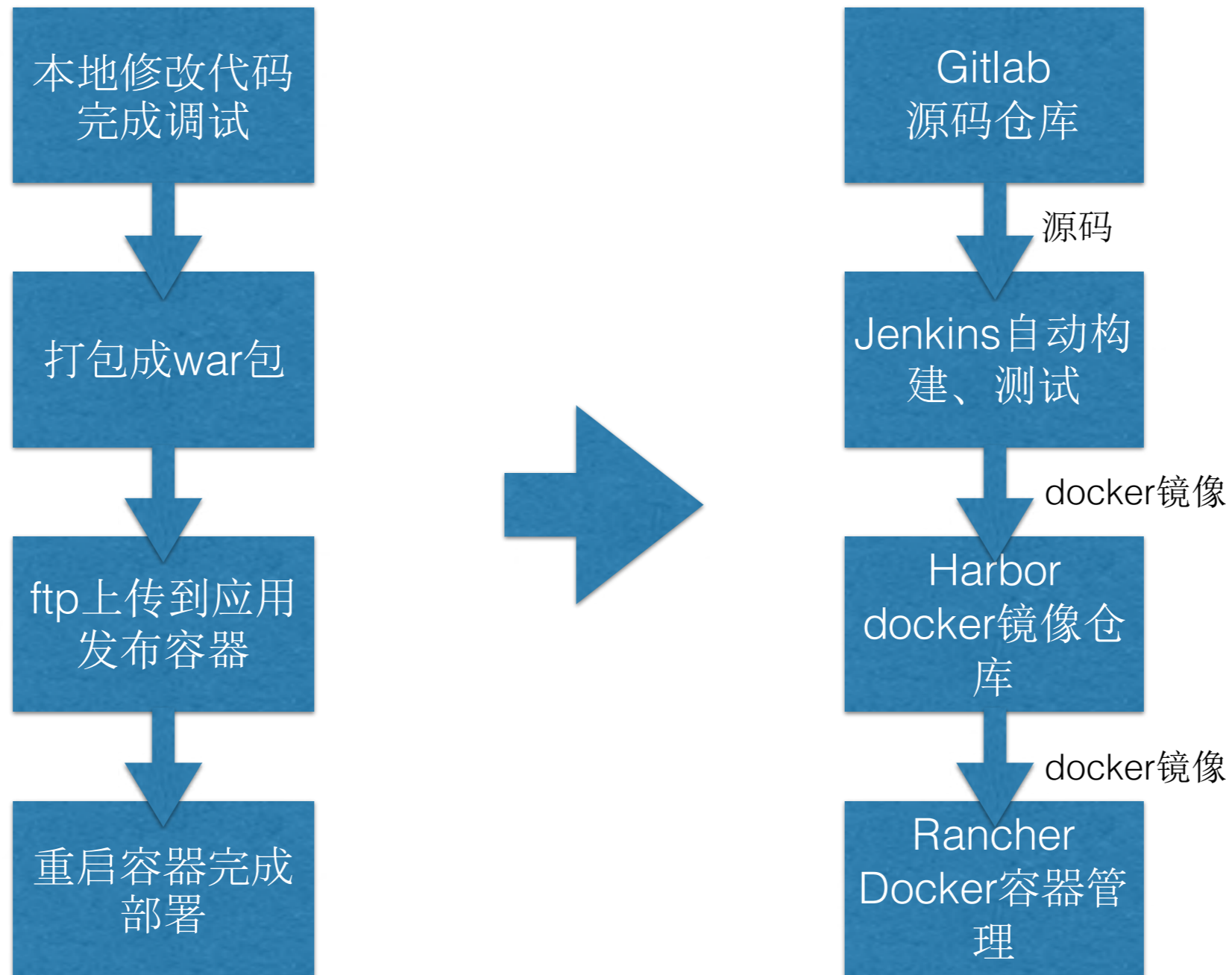
容器引擎

快速发布实践

与服务注册的结合

容器监控

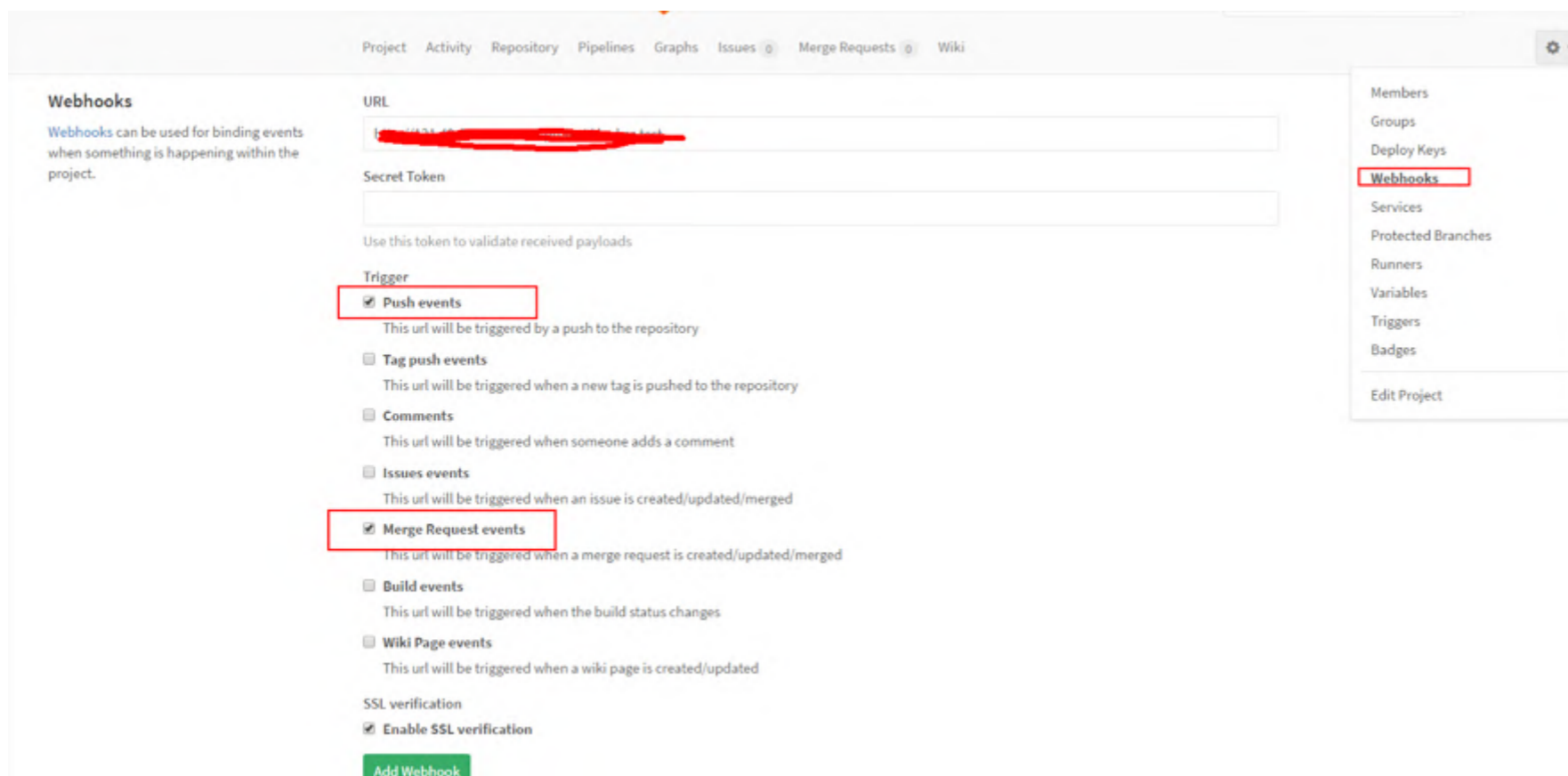
快速发布实践



Gitlab

- 源码仓库

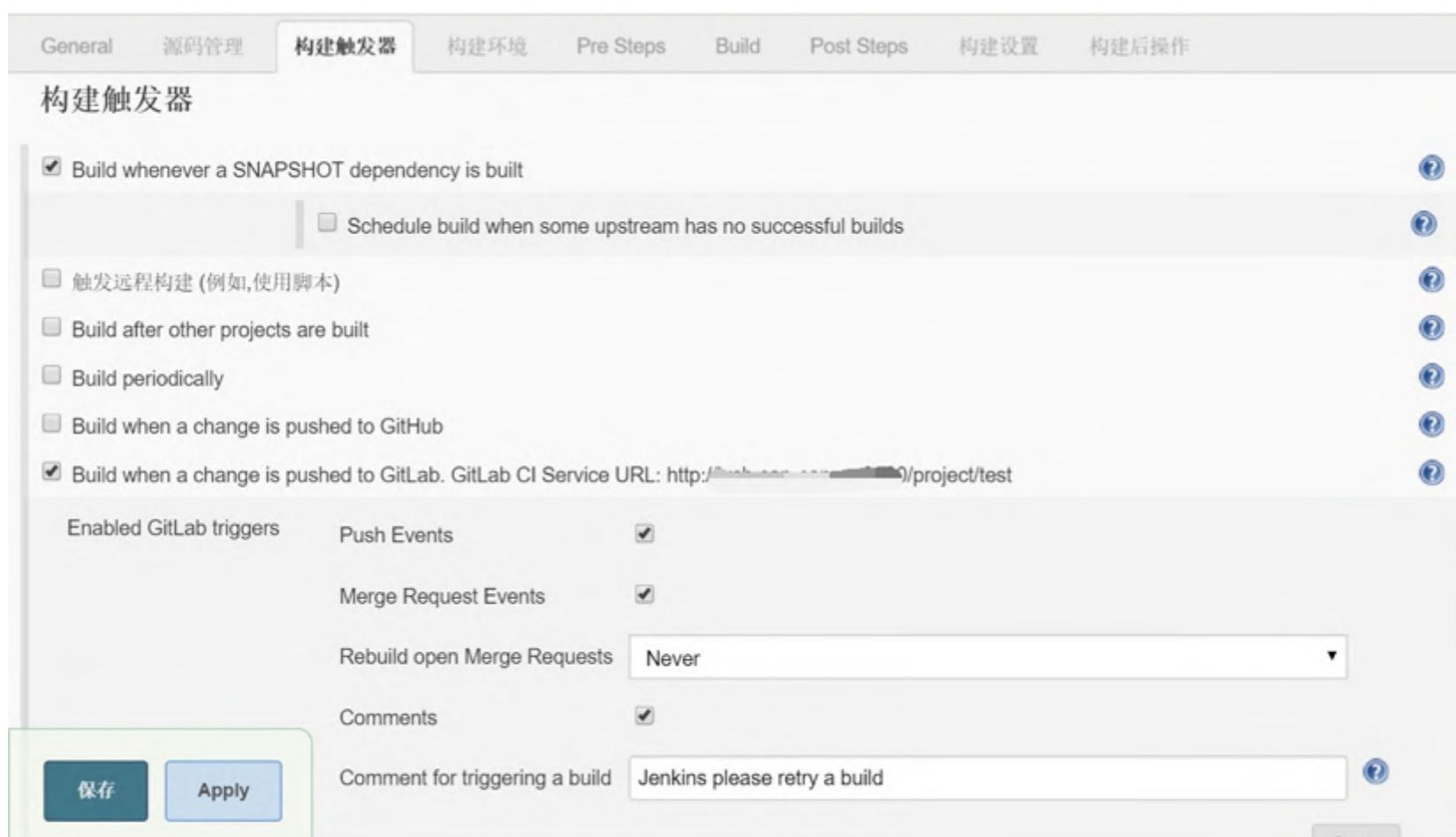
设置Webhook, 本地修改代码跑完ut, 提交到gitlab



Jenkins

- 自动构建

在每一次更新代码到git对应的分支后，jenkins将自动构建应用。

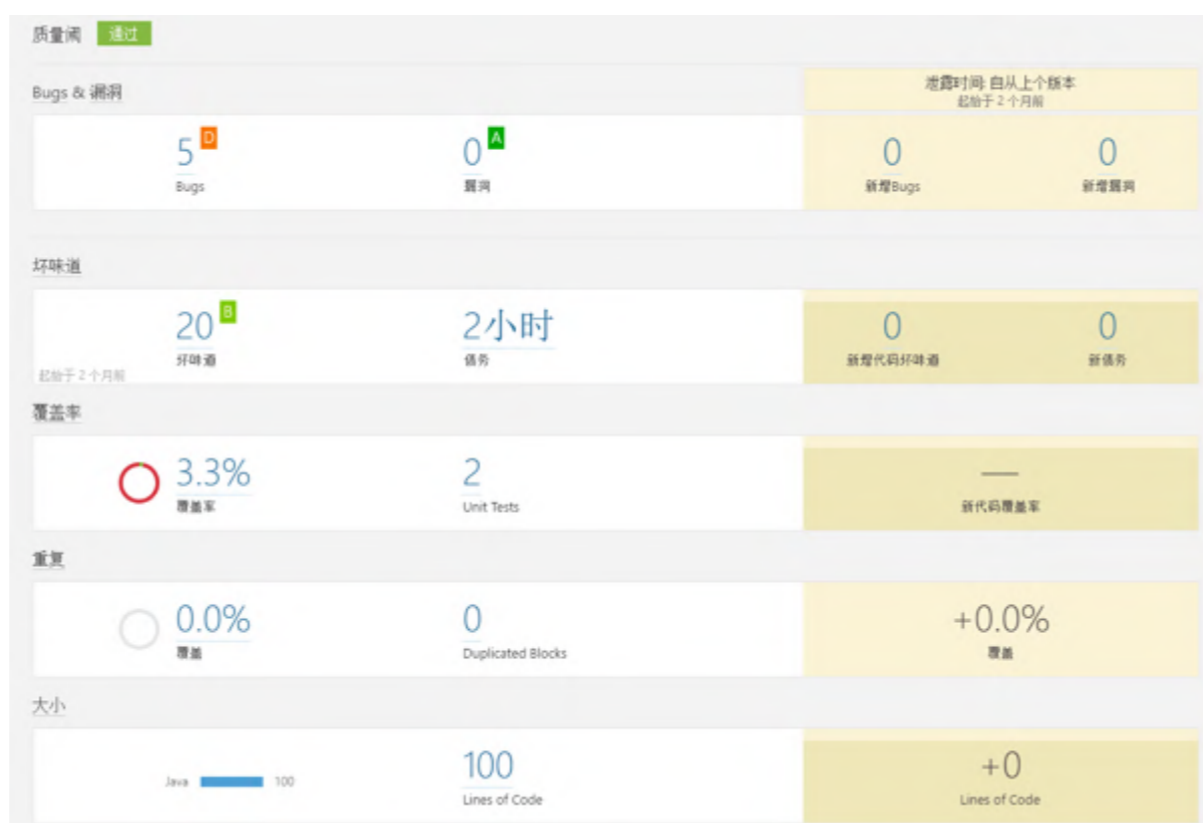


Jenkins

- 自动测试



此外，我们整合了sonarqube，完成每一次build后，还可以在SonarQube的服务
器上看见本次提交代码的质量情况



Harbor

- 镜像仓库



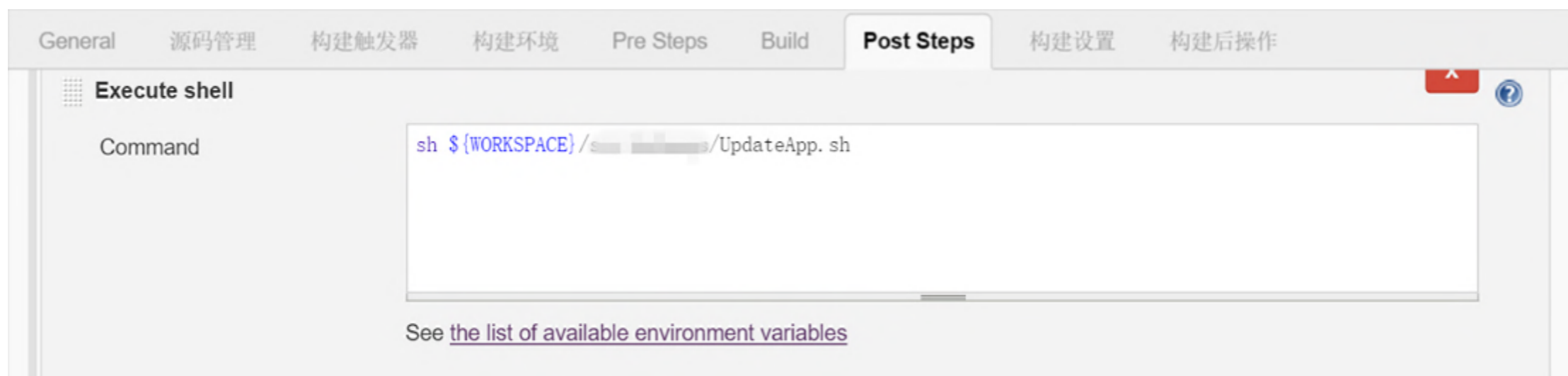
Harbor是可靠的企业级Registry服务器。企业用户可使用Harbor搭建私有容器Registry服务，提高生产效率和安全性，既可应用于生产环境，也可以在开发环境中使用。

A screenshot of the Harbor web interface showing the 'Post Steps' configuration panel for a Docker build. The panel is titled 'Docker Build and Publish' and contains several input fields: 'Repository Name' with the value 'cicdtest', 'Tag' with the value '\${BUILD_NUMBER}', 'Docker Host URI' with the value 'tcp://127.0.0.1:2375', 'Server credentials' with a dropdown set to '- none -' and an 'Add' button, 'Docker registry URL' with the value 'http://docker-registry-1:5000/', and 'Registry credentials' with a dropdown set to '- none -' and an 'Add' button. There are also 'Advanced...' and 'Add post-build step' buttons at the bottom of the panel. The top navigation bar includes tabs for 'General', '源码管理', '构建触发器', '构建环境', 'Pre Steps', 'Build', 'Post Steps', '构建设置', and '构建后操作'.

Ansible

- 结合脚本自动更新

在代码Build完成并将最新的镜像上传至仓库之后我们还需要自动的去触发Rancher更新应用，这里，由于我们去年年底做的时候，Jenkins还没有对Rancher很好的支持所以我们使用ansible来代替完成。其实就是在上传镜像那一步之后再添加一个Post steps去执行一个包含了Ansible命令的脚本，主要用于远程控制Rancher更新应用



Rancher

Rancher是一个图形化的集群管理工具，支持多种集群作为后端，包括Kubernetes、Docker Swarm、Mesos和Rancher内建的Cattle等。

Rancher

- Docker容器管理

The screenshot displays the Rancher management interface. At the top, there is a navigation bar with tabs for 'Production Env', '应用栈', '应用商店', '基础架构', '系统管理', and 'API'. Below this, the '用户应用' section is active, showing two application stacks: 'log-exporter' (日志转发器) and 'node'. Each stack has a '添加服务' (Add Service) button and a '排序' (Sort) dropdown menu. The 'log-exporter' stack shows 6 services, each with 1 container. The 'node' stack shows 3 services, each with 1 container. The services are listed in a table with columns for status, name, image, and service details.

应用栈	服务名称	镜像	服务类型	容器数量
log-exporter (日志转发器)	Service	镜像: [模糊]	Service	1 容器
	Service	镜像: [模糊]	Service	1 容器
	Service	镜像: [模糊]	Service	1 容器
	Service	镜像: [模糊]	Service	1 容器
	Service	镜像: [模糊]	Service	1 容器
	Service	镜像: [模糊]	Service	1 容器
node	cAdvisor	镜像: google/cadvisor:latest 端口: 18888	Service	1 容器
	exporter	镜像: prom/node-exporter 端口: 9100	Service	1 容器
	[模糊]	[模糊]	Service	1 容器

Rancher

- 主机管理



TABLE OF CONTENTS

向着微服务进军

演进中的问题

基于容器技术的解决方案

容器引擎

快速发布实践

与服务注册的结合

容器监控

与服务注册的结合

- 背景

- 服务启动时，需要向注册中心提供自己的信息，包括地址，端口，服务标识符等。然而在Docker容器中，服务一般无法了解自己在宿主机上开放的端口，也无法准确判断该服务的真实网络地址。这就给服务自身完成注册操作造成了一些困难
- 从另一个方面来看，服务所在的地址和端口等信息其实是和业务无关的，不应该也不适合由服务去处理。否则不仅背离了服务化的初衷，也加深了对服务的侵入

与服务注册的结合

- 基于容器的服务注册
 - Registrator通过监控Docker事件，根据docker容器的启动和停止向注册中心发送注册和注销的消息，无需服务自身参与即可完成。
 - Registrator能够掌握服务真正对外开放的端口。
 - 即使在服务无响应，甚至直接删除容器的情况下也可以完成对服务的注销。



TABLE OF CONTENTS

向着微服务进军

演进中的问题

基于容器技术的解决方案

容器引擎

快速发布实践

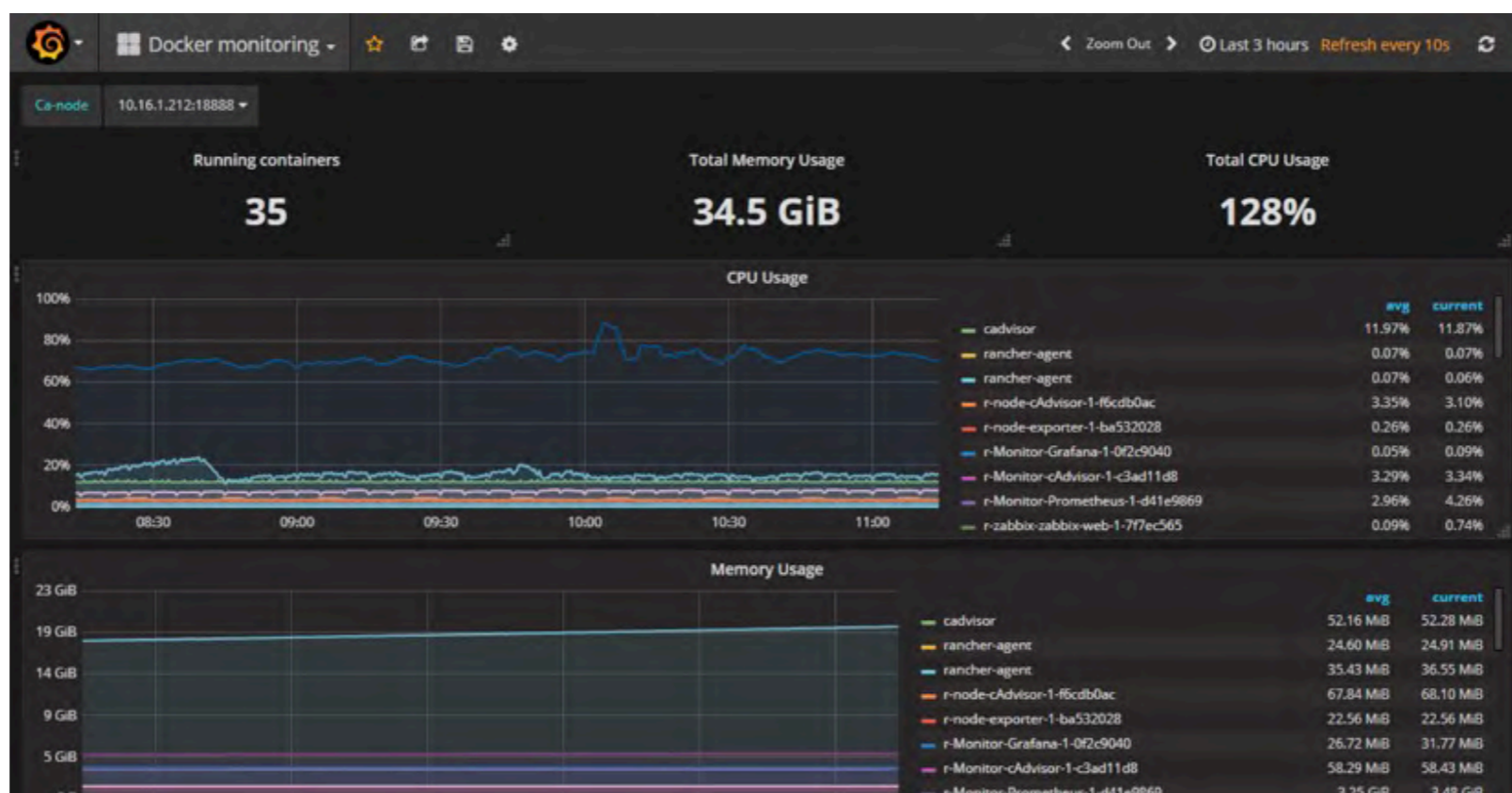
与服务注册的结合

容器监控

Prometheus

- 容器监控

使用了谷歌的cAdvisor项目作为prometheus的exporter。cAdvisor作为一个监控单机容器的项目，数据较为全面，结合prometheus后就能在整个集群监控查询容器的信息。



THANKS!

智能时代的新运维

CNUTCon 2017