干货分享 | 微服务配置中心架构解析

原创：IT老兵 岳晓阳 优云数智 6天前

本文根据优云数智技术总监岳晓阳于8月14日可信云大会《容器和微服务》论坛演讲整理而成，主要解析了配置中心在微服务的前世今生、微服务配置中心管理及原则、微服务配置中心Hawk架构解析以及未来展望，希望对大家有所帮助。

图：优云数智技术总监岳晓阳

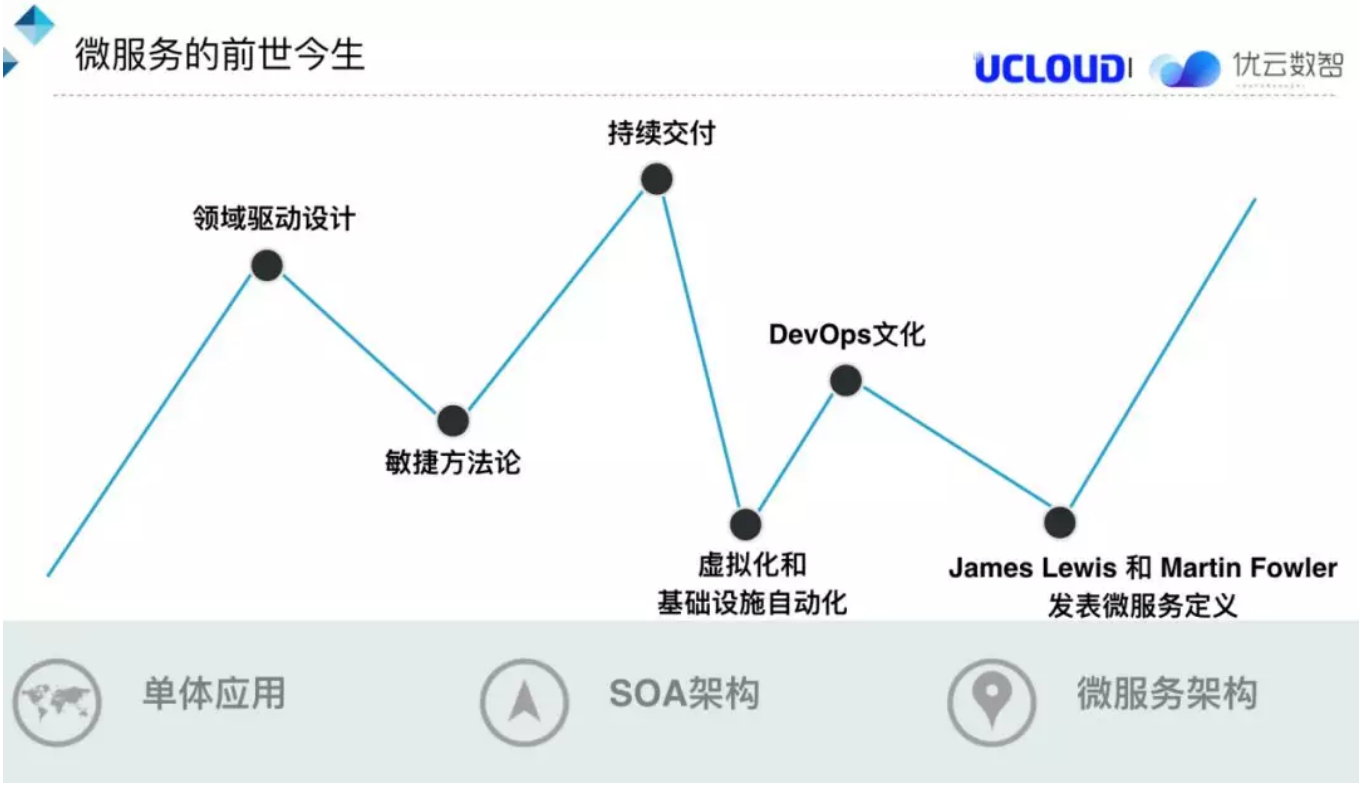
演讲提纲：

1.微服务的前世今生；
2.微服务配置中心管理及原则；
3.微服务配置中心架构解析；
4.未来展望

一：微服务的前世今生

配置中心在整个微服务体系里算其中一小块，只是解决了分布式环境下如何去做软件配置管理的问题。首先，先简单讲解一下微服务相关概念。

微服务本身的诞生并不是一个偶然的现象，从领域驱动设计、敏捷方法论、持续交付、虚拟化和基础设施自动化、DevOps文化这些因素都是推动微服务诞生的重要因素：



图一：微服务的前世今生

1. 领域驱动设计指导我们如何分析并模型化复杂的业务；
2. 敏捷方法论帮助我们消除浪费，快速反馈；
3. 持续交付促使我们构建更快、更可靠、更频繁的软件部署和交付能力；
4. 虚拟化和基础设施自动化(Infrastructure As Code)则帮助我们简化环境的创建、安装；
5.DevOps 文化的流行以及特性团队的出现，使得小团队更加全功能化。

软件开发经历的三个阶段：单体应用、SOA架构、微服务架构。因为本人做了差不多17年的开发，亲身经历了每一个阶段，在目前这个阶段，微服务现在基本上已经被广泛接受了。

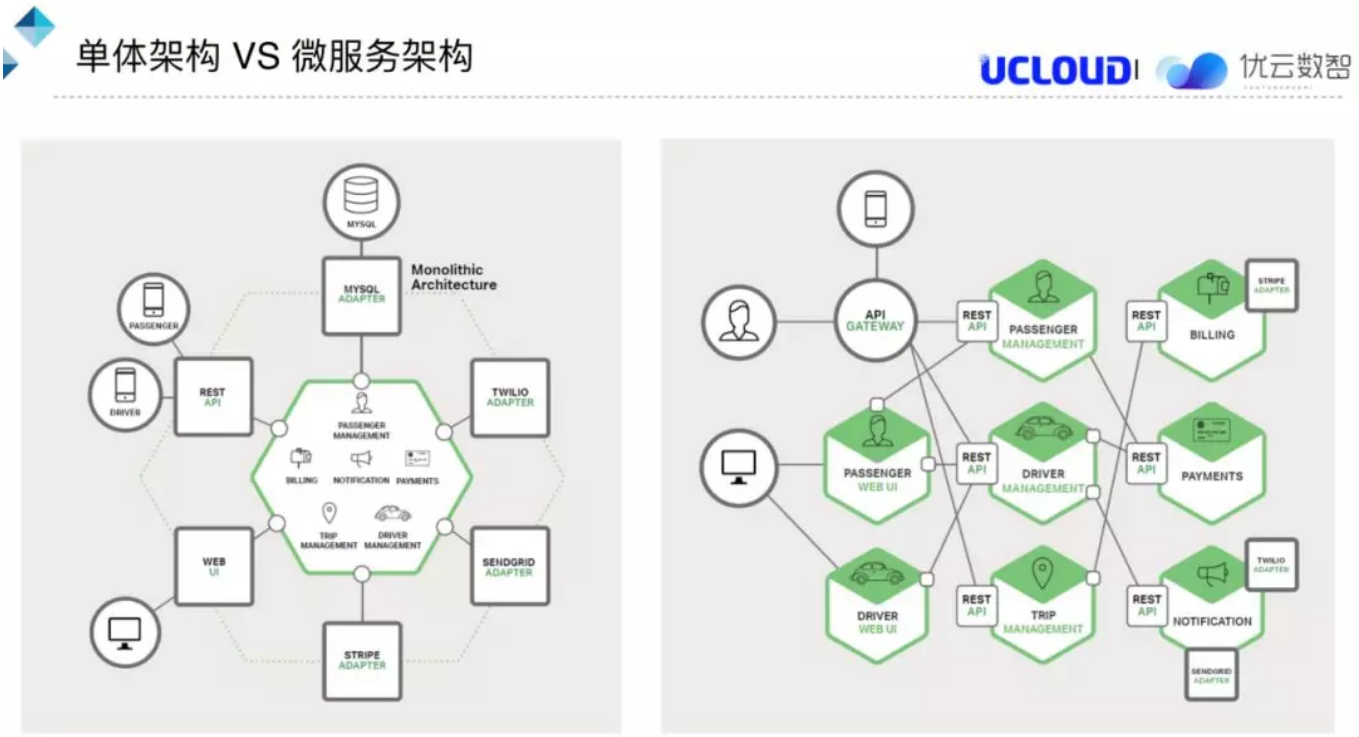


Figure 1: Monoliths and Microservices

图二：单体架构Vs. 微服务架构（来源：网络）

这里做一个比较，上图中两个架构图都是从网上下载的，实际是一个订车系统的架构，图中左侧是单体架构，右侧是微服务架构，从图上可以很清晰的看到两种架构的区别。单体和微服务的优劣对比，这里就不展开讲了，总的来说各有利弊，在一定的规模下，你很难说到底哪个好、还是不好。

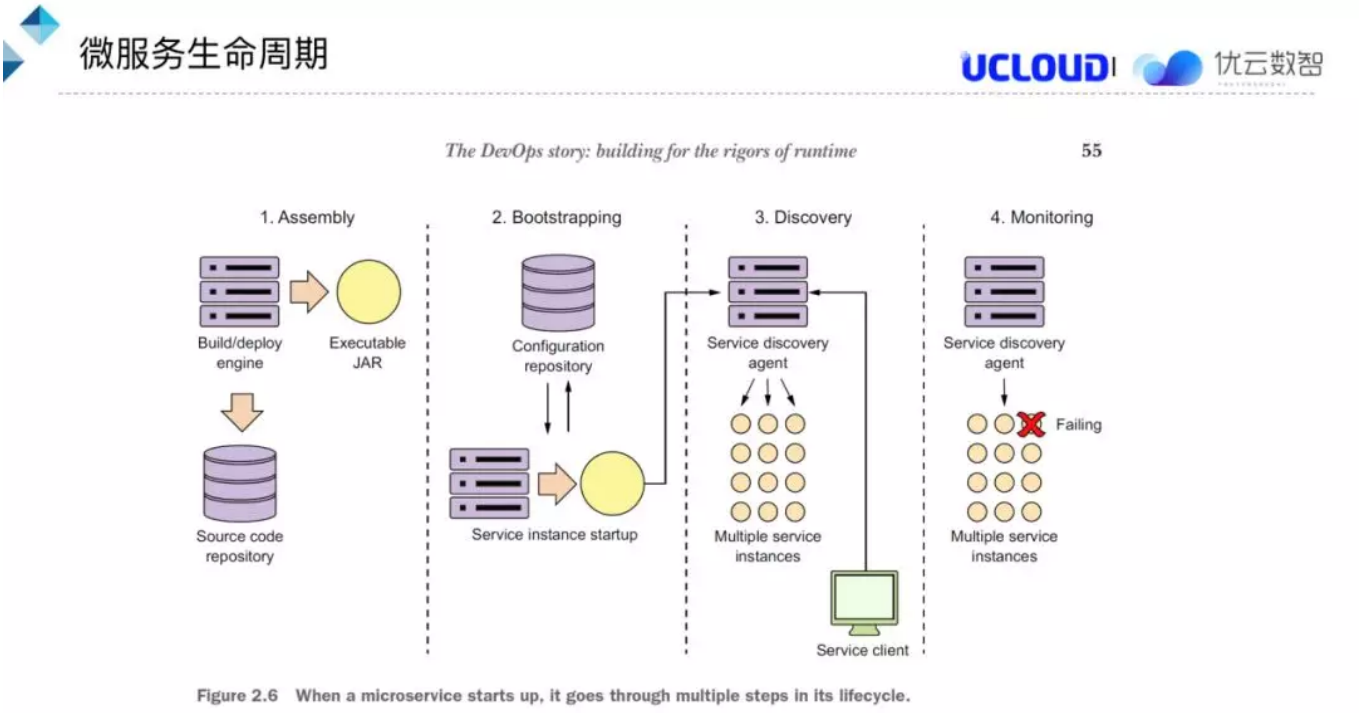


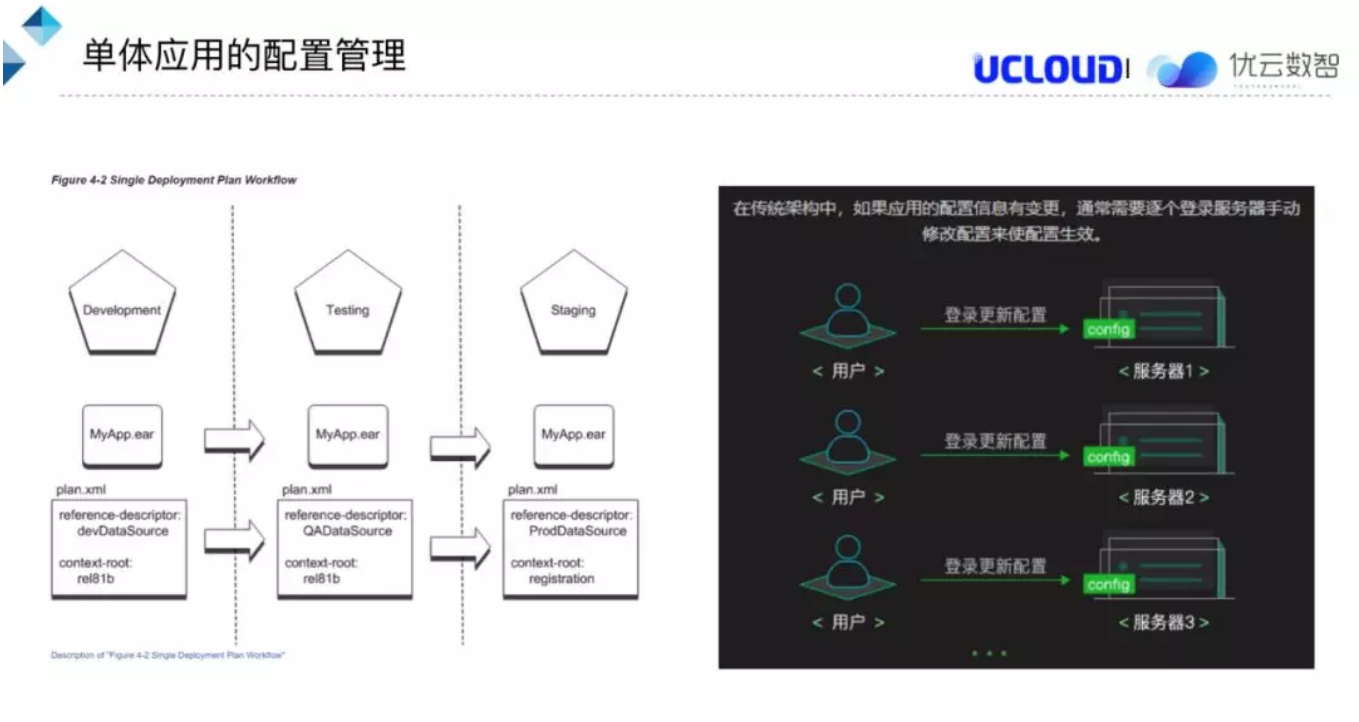
Figure 2.6 When a microservice starts up, it goes through multiple steps in its lifecycle.

图三： 微服务生命周期

从DevOps的角度来看，每次构建微服务并将其部署到环境时，都要经历组装、引导、服务注册/发现、服务监控四个阶段，分别对应了微服务开发的四个原则：

- 微服务应该是独立的，可独立部署多个
- 微服务应该是可配置的
- 微服务实例需要对客户端透明
- 微服务应该传达其健康状况

二、微服务配置中心管理及原则

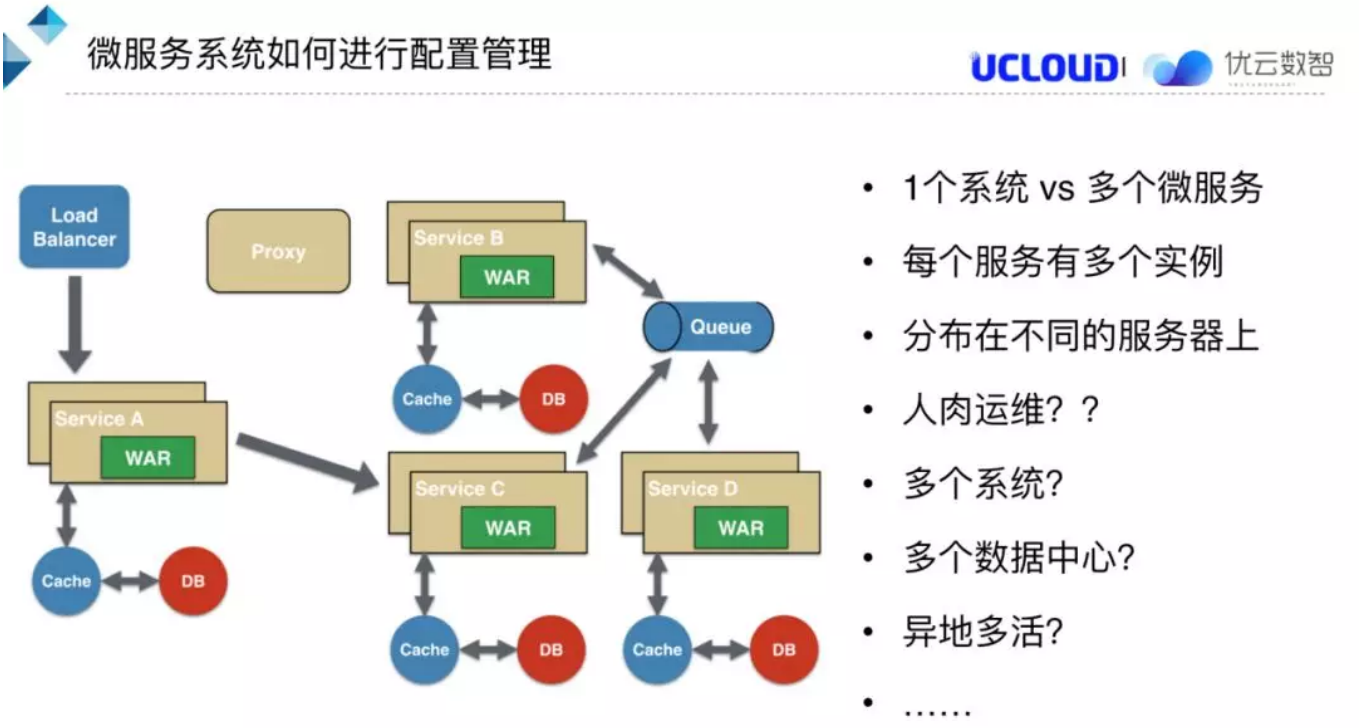


图四：单体应用的配置管理

我们认为每一个大型的分布式微服务系统都需要一个配置中心。从单体应用那个时候的配置管理来回顾一下，左边的图描述的是按照不同的环境，提供不同的配置文件，将这些配置文件跟二进制包打包在一起，在Weblogic时代这是很流行的一种做法。

到现在，有些解决方案提供商还是用这种方式来实现，它也是容器运行环境，在打镜像的时候会把配置的信息和镜像打在一起。假设，实际当中有三个环境，前面是一个开发环境，中间是测试环境，后面是一个企业的生产环境。每个环境都有相关对应的配置文件，程序去哪个环境运行，就提供对应环境的运行包。

2.1 微服务系统到底怎么管理配置信息



图五：微服务系统到底怎么管理配置信息

一个系统到后面可能会拆成好几个微服务，每一个微服务可能有好多的实例，这些实例分布在不同的系统上，要让人手工去改这个配置系统——这些都是构建微服务系统需要考虑的问题，想想就是很可怕的事。这还只是一个系统，如果是多个系统呢？牵扯到多个数据中心呢？所以对于微服务的配置，我们有几个原则：

图六：微服务配置管理

1.程序和配置一定要分离；

2.配置要集中进行管理；
3.同一个程序包要适应多个环境；
4.我们要提供一个客户端去拉取配置信息；
5.服务端要能够推送，这一条主要是考虑程序运行时动态修改配置的情况。

此外，还要维护多版本的配置信息、配置中心自己的容灾以及客户端规模达到一定数量的时候，必须考虑配置中心的性能问题。

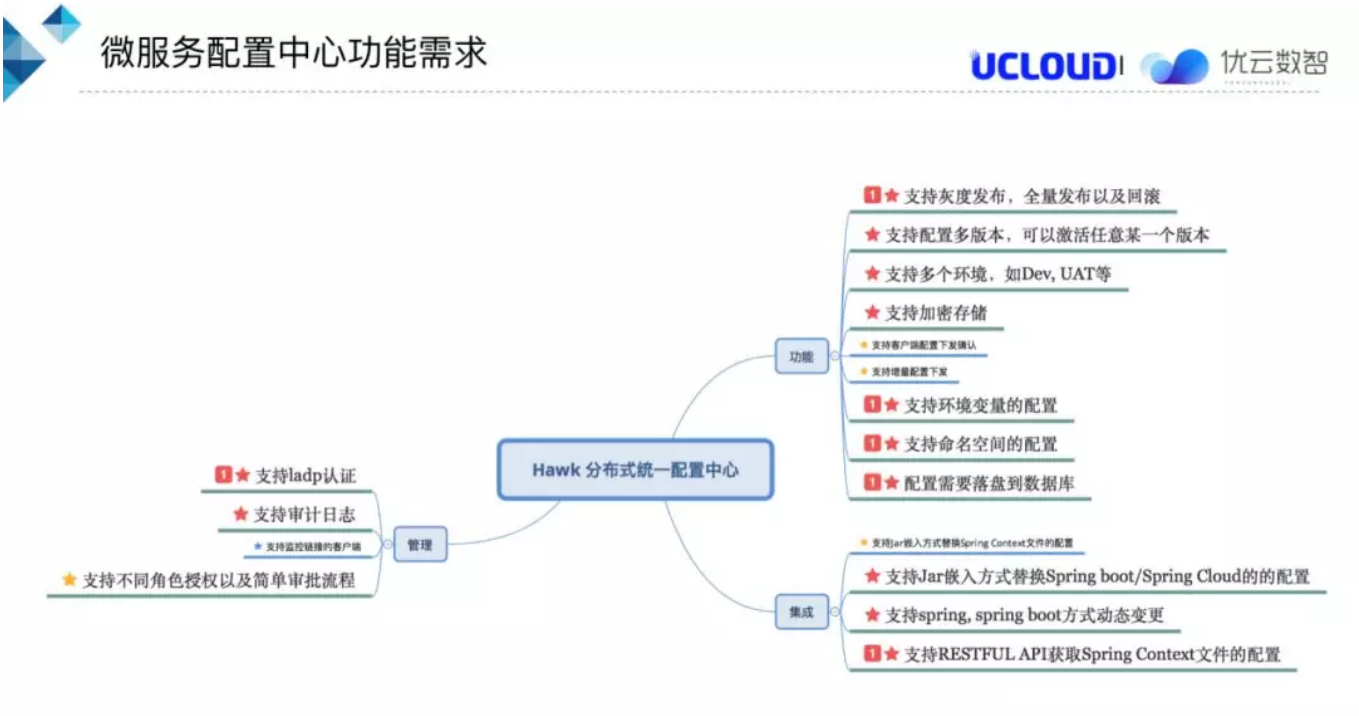
2.2 微服务配置原则

Heroku创始人AdamWiggins发布了一个“十二要素应用宣言（TheTwelve-Factor App）”，为构建使用标准化流程自动配置，服务界限清晰，可移植性高，基于云计算平台可扩展的服务配置提供了方法论：
1. 配置是可分离的，可从微服务中抽离出来，任何的配置修改不需要动一行代码；
2. 配置应该是中央的，通过统一的中央配置平台去配置管理不同的微服务
3. 配置中心必须必须可靠切稳定地提供配置服务；
4. 配置是可追溯的，任何的配置历史都是可追溯，被管理且可用。

在云服务时代，对微服务做配置，对它有什么样的要求呢？首先，必须基于镜像管理部署，有自己相应独立的配置，而且程序包不可以因为环境的改变而更改。也就是说，它是独立于环境的不可变的程序包。**这是我们提到的，云化微服务的配置原则**：

1.完全分离要部署的程序和其对应的配置；
2.程序包对于任何环境都是不可变的；
3.通过环境变量或配置存储在程序启动时注入配置。

2.3 配置中心功能需求分析

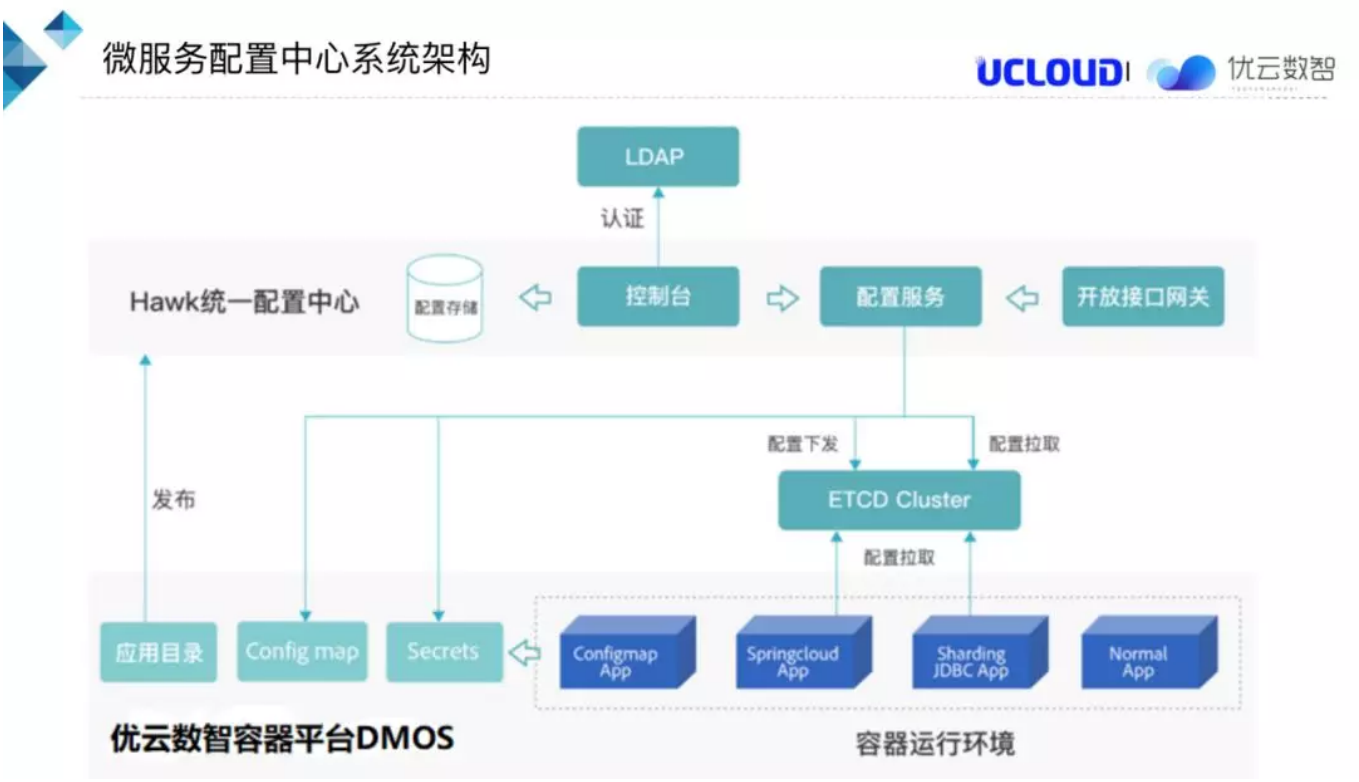


图七：微服务配置中心功能需求

上图是我们对配置中心功能需求分析的整理，主要分三个大的方面：一是需要具备的功能，二是跟其它产品的集成，三是企业级的管理属性。我们认为配置中心应该具有以下四个必备要素：

1.配置数据持久化存储。
2.可以横向扩展的缓存集群。
3.配置信息的拉取和推送。
4.配置数据及配置过程管理。

三、配置中心Hawk系统架构解析

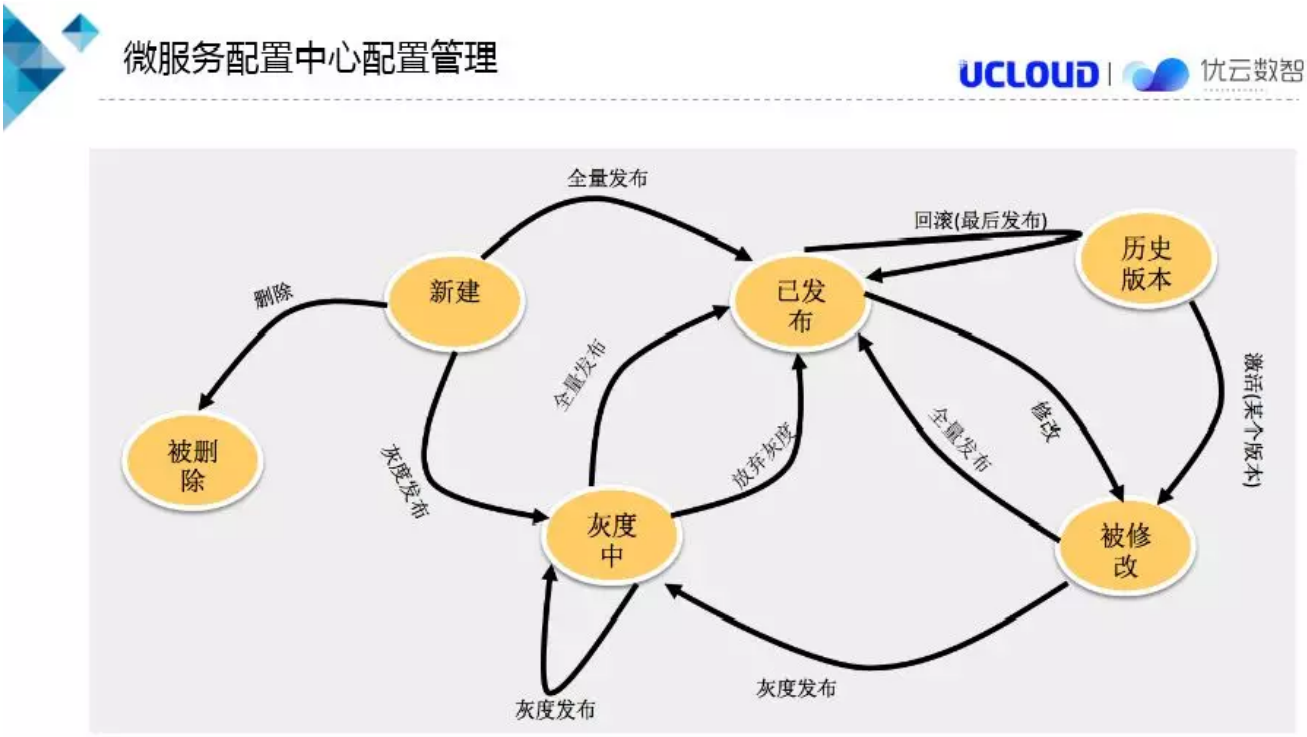


图八： 微服务配置中心架构解析

首先，接入第一层是网关，整体的存储通过Hawk Server，下发到ETCD集群，ETCD集群再同步到K8S容器运行的平台。先从数据迁移的状态简化成简单的几部分。比如新建一个配置，要么配置就被删除了，直接一步到位。如果没有这样做，就面临几种情况：

1. 这个配置是否要小范围的去做一些试探性的发布，这种情况可以走灰度发布，状态变成灰度中，配置不允许更改。要么就是两条路走，全量发布到所有服务上。要么就是放弃灰度回到之前的状态，放弃灰度后会去到已修改的状态。

2. 另外一种情况，新建一个配置，直接全量发布，状态变成已发布状态，这时候是可更改的。但是每一次的更改，还是会回到原来那个状态。这个更改要做灰度吗？还是做发布？还是对发布有点后悔，不打算更改了？这时，从历史版本里面找一个合适的版本，激活，然后再做一次发布，通过几个简单的回路，涵盖了大部分的业务场景。

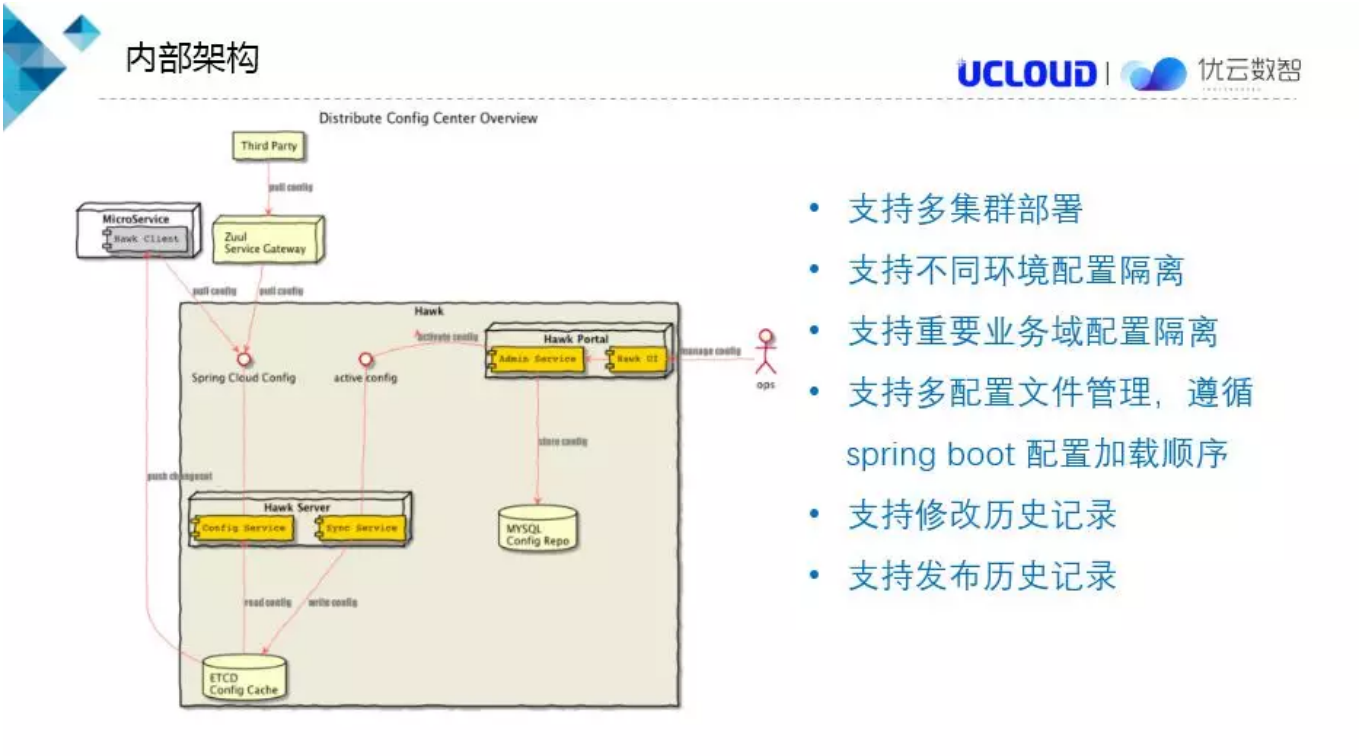


图九：微服务配置中心配置管理

3.1 配置数据状态变迁

Hawk Portal是主体的配置界面，用户在界面上对配置进行输入、增删、改查的管理。这些资料会有两份，一份做通过Mysql做本地存储，另一份通过Hawk Server直接同步到ETCD。

由于HAWK Server是同步到ETCD里面，也就是说ETCD相当于另外一个数据库，这当中不存在数据之间的互相抄送，从而减低丢失数据的风险。持久化，是说研发和运维在后台做数据迁移，或者数据监控时更有把握，更方便。

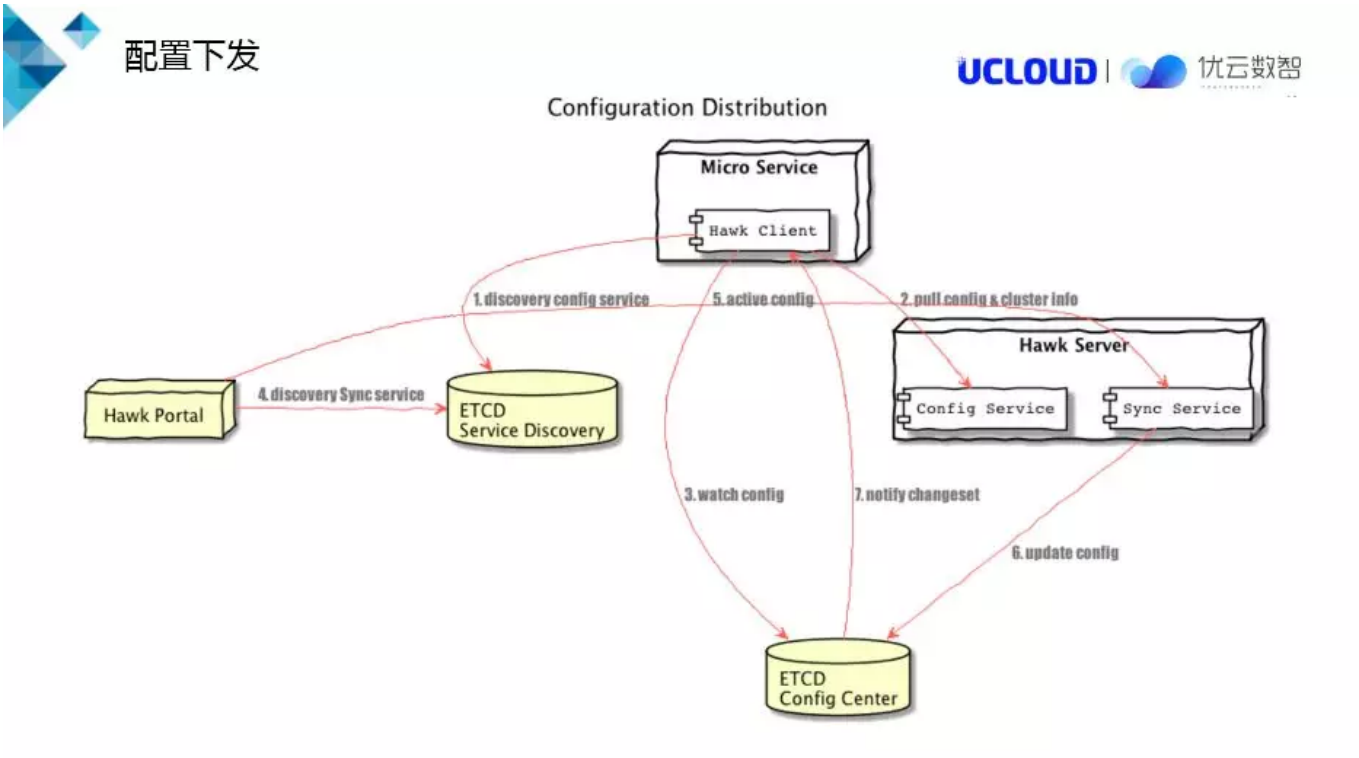


图十：内部架构

优云数智HAWK其实有两个ETCD，一个ETCD是做注册发现的，Hawk Server、Hawk Portal都会注册在里面，作为相关的组件。类比Spring Cloud Eureka，Eureka是注册在Eureka Server里面的一个内存列表，集群里面所有Server共享这个内存信息。这个过程优云数智做了简化，所有信息全部注册在ETCD里面。

ECTD集群由于是共享的，组件的状态和一致性得到保障。Portal和Server之间不再通过Portal注册在Server并通过心跳来维持关系而是通过共享持久化的ETCD，保证数据在任意时刻所看到的状态都是一致的，从而保证了服务的注册，以及服务发现的稳定性。

Hawk和Eureka 选择的路径不一样。Eureka是比较重量级的，HAWK则简化了这个配置，简化这种代码的复杂性，重点提高系统的完整性，打造系统闭环，通过一些相对简单的方法，提高服务的稳定性。



图十一：配置下发

配置一旦通过Hawk Portal潜入本地数据库，微服务的注册服务是怎么实现配置呢？当Portal写入配置到本地数据库时，同时也会通过服务Sever去同步到ETCD，ETCD里面存储的信息，是一个持久化的数据。

通过Server实时从ETCD拉取配置，有时是运行的时候拉取，有时是启动时拉取。启动时拉取有两种策略，启动的时候拉取配置，存储到本地作为静态文件的配置，运行时候拉取，动态的变更实时生效。

在Web层其实也有一些问题需要解决，比如，因为我们不是一个开发框架，是奔着一个开源系统的方向去，所以要解决服务跟浏览器之间的授权。

优云数智现在的做法是在本土数据库存储一些用户的信息，但是并没有采用传统意义上的建Session来做验证和授权，而是通过动态下发JWT的形式，每一个请求动态下发，根据我个人用户的一些信息生成，每次的请求一来一去都有交换新的Token，每个Token实时生效并有续约的功能，来代替传统意义上的Session。

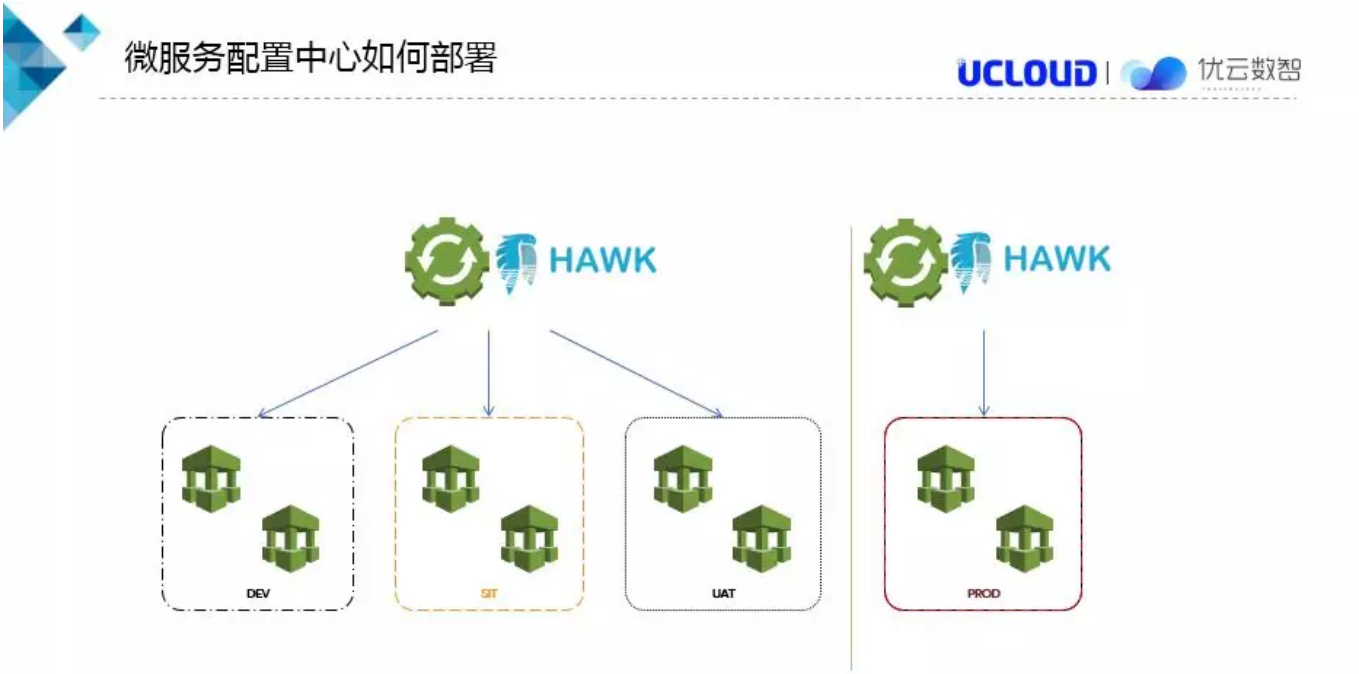
3.2 配置中心的支撑体系

第一种运维管理体系类似于偏静态类的配置，在启动时通过配置文件直接拉取读业务；另外一种是开发管理体系，偏动态管理，代表的是一种程序或者在运行过程中，通过实时的变更配置内容而实时生效，达到的一种效果。一个健全的配置中心应该支持这两种运维体系。配置中心应该具备有以下几点特性：

1.基于Spring Cloud config打造。
2.完全兼容Spring Cloud config API。
3.配置更新通过GRPC双向流实时推送。
4.采用ETCD作为配置数据的强一致性存储。
5.具备LDAP用户认证、授权管理、审批流程、审计日志等企业级特性。
6.通过Open API和查件体系扩展支持基础组件，如sharding-gdbc。

3.3 配置中心部署模式

微服务配置中心如何部署，一般来讲会是这么两种方式，生产环节中单独布一套配置中心，在开发环境部署一套配置中心，开发环境的配置中心可以支持Dev、SIT、UAT等多个环境。



图十二：微服务配置中心如何部署

四、对未来的展望

我们要做的工作，首先还是继续围绕Spring Cloud体系，为用户提供成熟方案和服务治理中心；二是探索基于Service Mesh的新方案，拥抱Istio/Conduit；三是配置中心Hawk我们已经做好了在GitHub上开源的准备工作，希望大家多多关注Hawk，关注您可信赖的云服务合作伙伴——优云数智。

图十三：未来展望

讲师介绍

岳晓阳，17年IT老兵，曾长期负责电信系统开发和互联网架构技术开发工作，具有丰富的大型软件系统技术架构和项目实施经验。对微服务、分布式系统、容器化等云计算领域的相关技术都有深入研究。