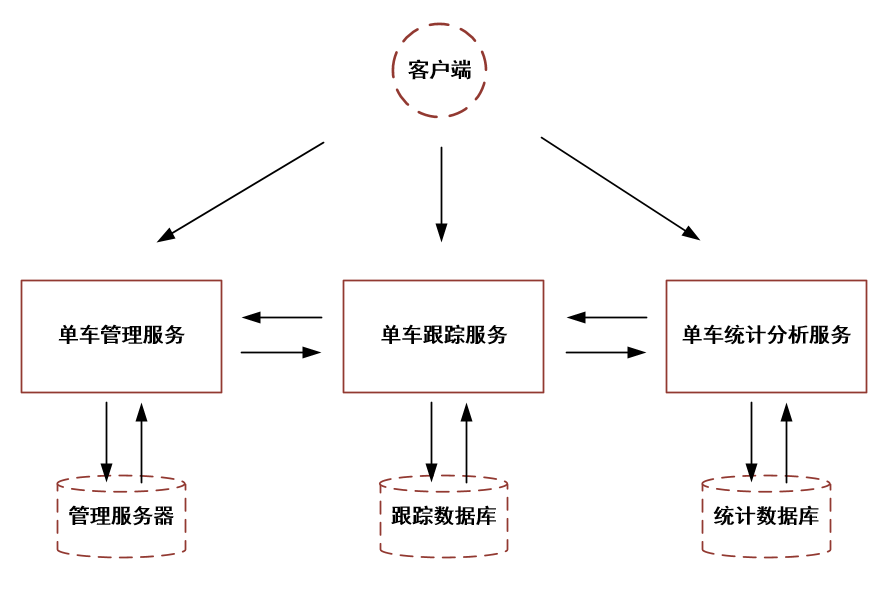
共享单车微服务搭建

金天阳 刘小兵 黄建伟 李志辉

我们搭建的共享单车是一个概念验证型（proof-of-concept）应用，通过使用Spring Boot, Spring Cloud 和 Docker，用简洁的用户界面对微服务体系模式（Microservice Architecture Pattern）进行论证和呈现。

**功能服务**

共享单车应用可以被分解成三个核心的微服务。这三个微服务是独立部署的应用，围绕特定的业务能力进行组织。



**单车管理服务**

单车管理微服务主要实现的功能是“新增单车录入功能”，当需要向市场中投放新的共享单车的时候，管理员会通过单车管理微服务实现新增单车的录入工作，使新的共享单车加入到整个应用管理当中来。

**单车跟踪微服务**

单车跟踪微服务主要实现的是单车位置的跟踪服务，分成两个方面的功能：

1. 上传坐标点。用户可以在使用完单车后上传单车的位置信息，以供以后其他人使用，或者方便单车管理人员的维护工作。
2. 获取坐标点。用户或单车管理员可以通过客户端获取单车跟踪微服务中的单车位置信息。

**单车统计分析服务**

单车统计分析服务主要实现的是对单车位置、数量以及使用情况的统计分析功能，管理员可以利用此微服务去实现对共享单车的合理调度、维护和数据统计。

**要点分析**

* 每个微服务都有自己的数据库，不使用API将无法直接访问持久化数据。
* 本工程中，我使用MongoDB作为每个微服务的主数据库。也许支持多种持久性结构显得更有意义（可选择数据库的类型是服务最适合的方式）。
* 服务之间的通讯非常简单：微服务只使用同步的REST API进行通信。现实情况中，常见的做法是结合使用不同的交互方式。例如：为了解耦服务和缓存消息，GET操作中一般使用同步的方式请求和获取数据；而在创建和更新操作中则通过消息代理实现异步处理。这些方法将达成最终一致（eventual consistency）的目标。

**基础架构自动化**

相比部署单个的应用程序，部署微服务及其相互依赖性要复杂得多，拥有完全自动化的基础架构显得非常重要。我们可以通过持续交付方法获得以下益处：

* 随时发布软件版本的能力
* 任何构建结束都可以发布
* 一次构建-根据需要部署