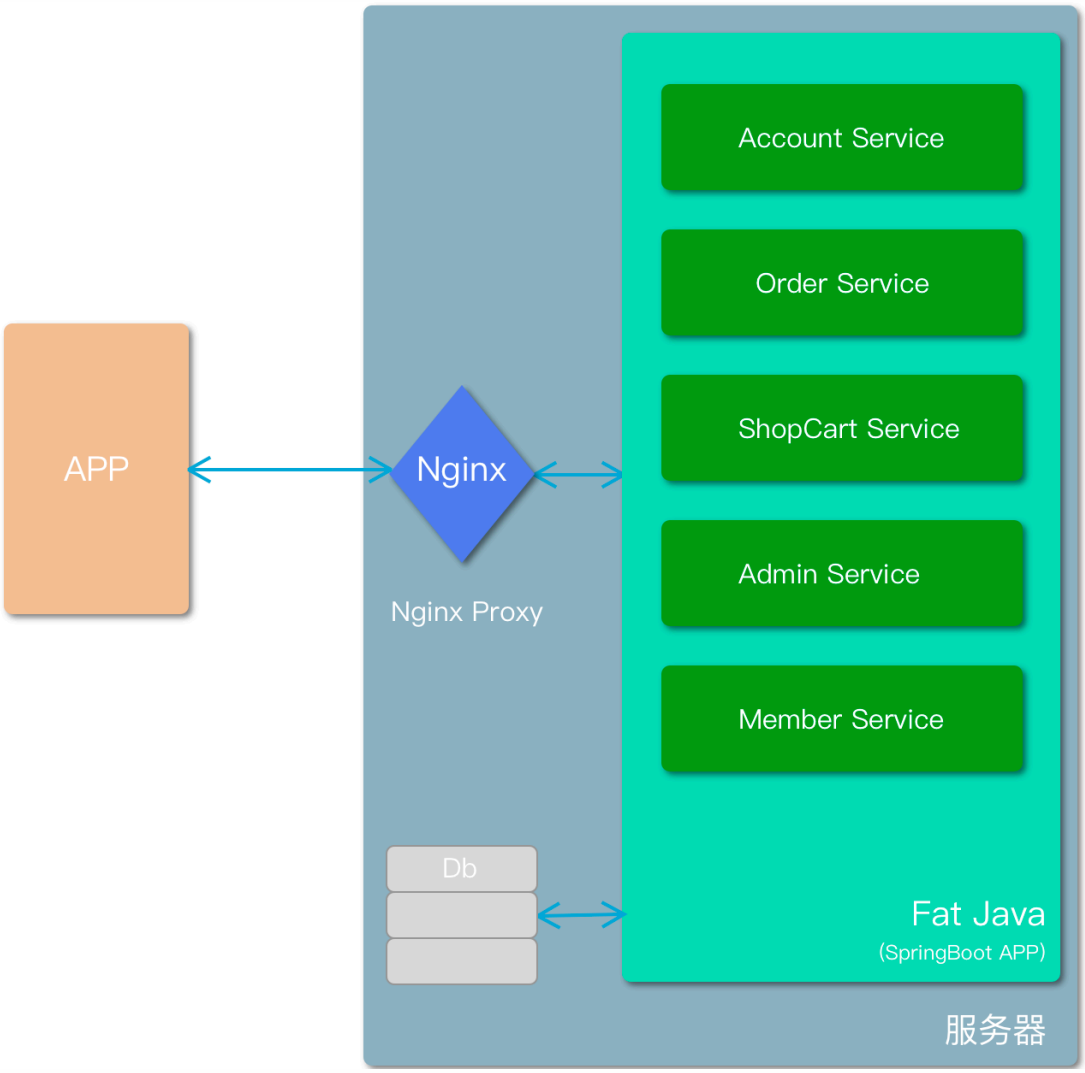
一、单体应用与微服务

什么是为微服务？，在众多后台开发人员和运维人员中，十个有九个都听说过微服务，但能真正说清楚的什么是微服务的却很少。有人说，微服务是一种架构，也有人说微服务是一种编程模式，各人有各人的说法，每个人也有每个人的理解，就像是一千人眼中就有一个哈姆雷特一样，微服务俨然成了与哈姆雷特一样的存在。但在真正的实践中，微服务架构，即表示一种架构上的风格，也用于开发人员的组织划分，二者相辅相成，互为阴阳。

在微服务兴起之前，开发软件大多都采用单一应用的模式，即使用语言提供的打包或分发工具，构建单一的 `APP`，放在服务器上去运行，众多开发人员维护一套代码，一个程序员犯错就会导致整个应用无法构建，这在记录片 代码奔腾 里面提现的淋漓尽致，该片有个片段讲述了 `Mozilla` 在开源发布会前期，因为一个程序员，犯了一个错误，导致整个 `Mozilla` 浏览器无法 `build` 成功，险些影响了开源发布会。在 `build` 情况下犯错，算是比较轻量的，因为在没上线之前的错误，都可以及时进行补救，如果是运行时错误，那对整个应用将是致命的，一旦出错，将导致整个服务不可用。

单体应用虽然也是多模块开发，但最终还是会打包并部署为单体应用。最后这个应用因为太过复杂，以至于单个开发者都无法理清整个应用，最终导致，应用无法扩展，可靠性低，开发效率低下，严重时可能导致整个应用无法完成交付。

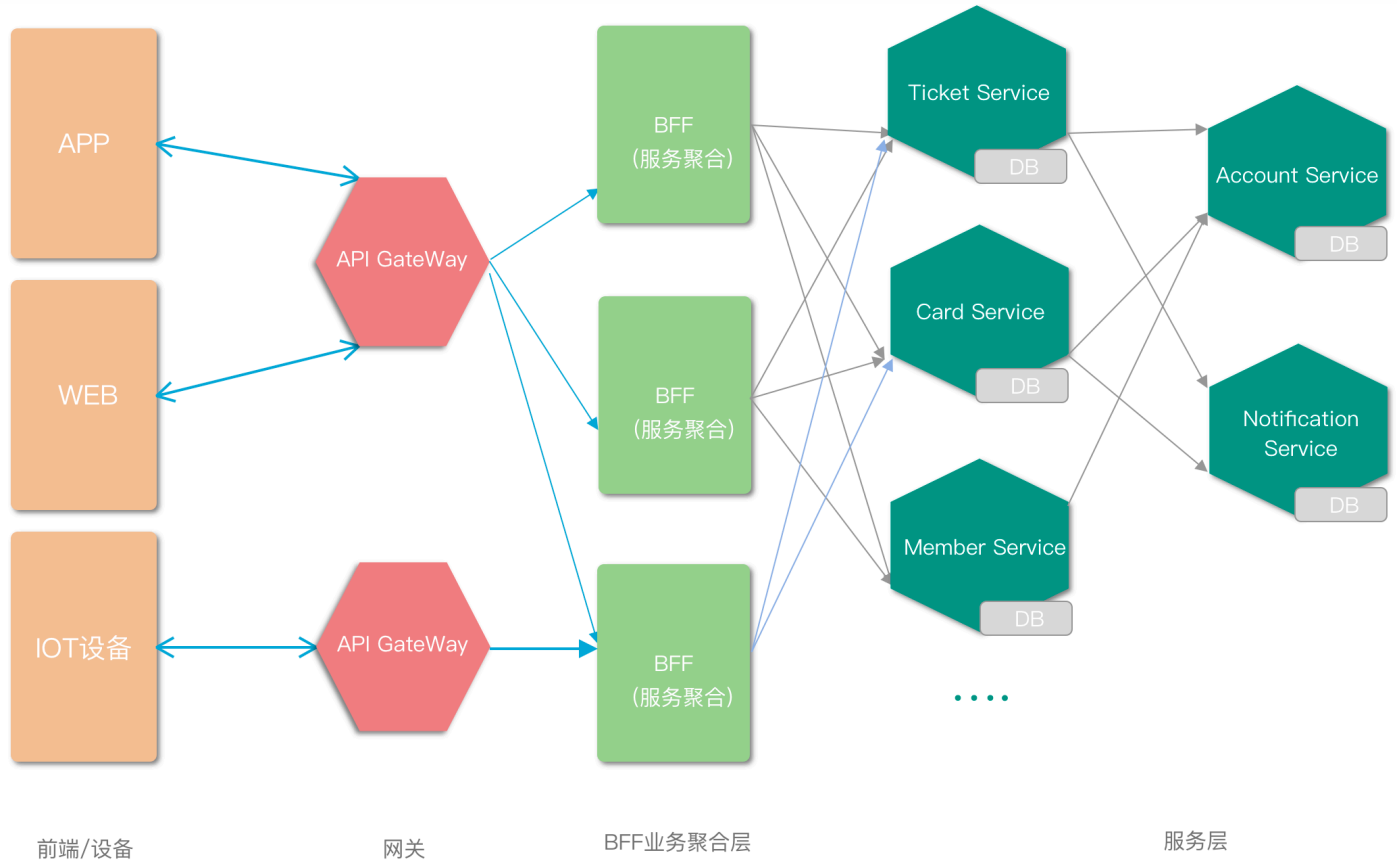
> 常见单体架构



微服务与单体应用不同，微服务首重要点在于 微 字，在上图我们可以看出，单体应用是将所有业务服务，全部打包成一个 `JAR` 包中，然后部署到服务器上，再通过 `Nginx Proxy` 代理出来， `Nginx` 还需要对静态资源做一些处理。微服务则是将单体应用里的服务，先拆分，再聚合。为什么说先拆分再聚合呢？拆分就是将单体应用中的 `Service`，按业务域或功能进行拆分成一个个单独的服务，每个服务都可以单独进行部署，都是一个 `Fat`

`Java` 应用，服务之间是进程隔离的。而聚合，就是将拆分出来的服务功能，进行业务数据组装，以供 `APP`，`WEB`，`嵌入式设备` ,这些前端应用使用。

> 微服务简单示意图



从微服务的示意图上可以看出，微服务架构风格下的应用，远比单体架构风格下的应用复杂很多，涉及的技术方案也是单体架构风格下的应用远不可比的。示意图只是最简单的架构，在实际架构中，涉及的方方面面远比图上的还要多，上图就忽略了负载均衡这一层，在 `API Gateway` 与前端之间，还有一层 `LB`，可以是硬件也可以是软件。一般常用的是 `Nginx`，或是 `F5` 这一类的硬件负载设备。

微服务之于单体架构，在上图我们可以看出，我们将整个服务细化了，根据业务分出了不同的微服务应用，应用之间的通讯，一般会采用 `RPC`，或类似 `Double` 这一类框架的自有协议，考虑兼容或多语言环境，一般推荐 `gRPC` 作为服务之间的通信，方便多语言服务之间进行通信，与采用的底层框架进行解耦。每个微服务都独享一个 `DB`，都是独立可运行的一个应用，其实只是 `DB` 是独享的，缓存也是每个服务独享的，做到服务与服务之间的数据隔离。

二、微服务基本概念

微服务是围绕业务功能构建的，每个微服务关注的是单一的业务，服务与服务之间，采用轻量级通讯机制，可以全自动独立部署，并且可以使用不同的编程语言，数据存储技术，微服务架构通过业务的拆分，实现服务组件化，通过组件组合快速开发系统，因为每个服务都可以独立部署，所以使得整个系统将变得清晰灵活，可以做到服务与服务之间的松耦合。

三、微服务基础设施

1. API Gateway (服务网关) —> SpringColud Gateway，OpenResty, Kong, Zuul
2. RPC —> gRpc服务之间通信
3. 服务发现 —> Euerka，Nacos，Etcd, Zookeeper, Consul
4. 消息中间件 —> Kafka, RocketMQ, RabbitMQ

5. 容器服务 —> Docker, Kubernetes(K8s)

四、微服务优点

1. 原子服务
2. 独立进程
3. 隔离部署
4. 去中心化服务治理

五、微服务缺点

1. 基础设施建设，复杂度高
2. 服务拆分粒度不好把控
3. 对测试不友好
4. 对运维要求比较高

结语

总的来说，微服务对开发和运维都提出了比较高的挑战，但后期收益也是非常可观的，使整个产品开发可控，稳定性提高，职责划分明确，能快速定位到相关人员，可以快速响应处理，微服务集群，还可以解决单个服务发生故障，不至于整个服务不可用，提高了整个服务的高可用性。