微服务架构是一个用分布式服务拆分业务逻辑，完成解耦的架构模式。（就是把以前的Bll改成服务，跨进程调用服务来完成功能）

SOA：面向服务架构：是一个组件模型(通常与微服务混合使用), 它将应用程序的不同功能单元(服务)进行拆分，并通过不同服务之间定义良好的接口和规范联系起来

SOA和微服务区别:

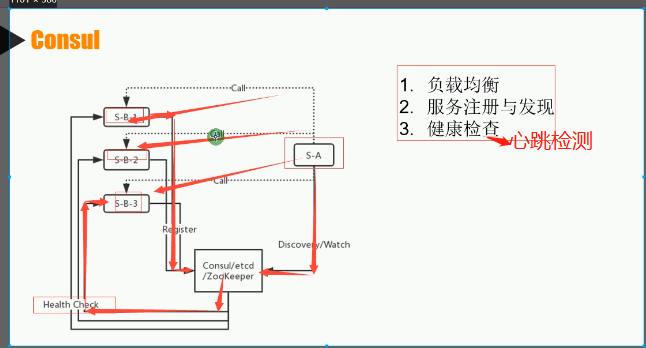
1. SOA是为了重用，微服务是为了重写
2. SOA更水平，微服务垂直
3. SOA自上而下，微服务自下而上

微服务------进程之间通信：

1. 共享存储： Redis,DB,Queue,硬盘文件等等，被动式通信，门槛低。
2. 服务通信： WebSoeket, WebService,WCF,WebApi,ashx,aspx等等，主动触发，数据序列化传递，跨平台，跨语言,http穿透防火墙.
3. RPC： .net Remoting(.net独有，不支持跨平台);gRPC(高性能，开源，通用的RPC框架，面向服务端和移动端，基于http/2设计)

微服务------服务实例集群：

服务的高可用，可伸缩

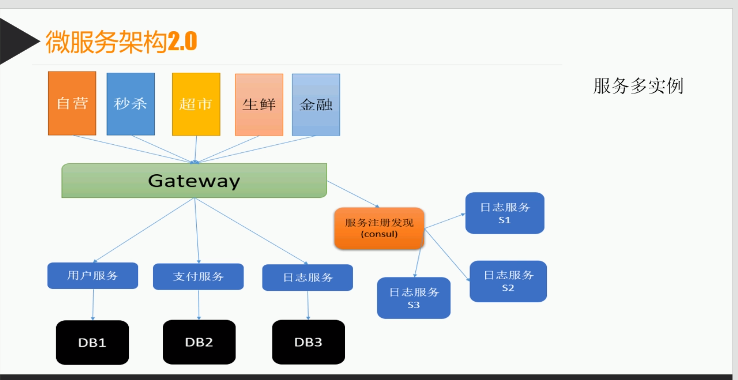


微服务------网关：

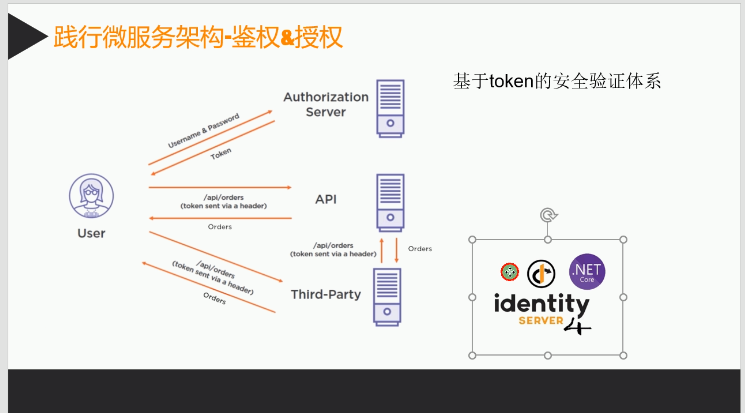


微服务------网关GateWay服务治理：

1. 缓存
2. 熔断：单位时间内异常多少次，就指定时间内，网关直接返回失败。
3. 限流：单位时间内，流量只允许多少次。
4. 降级：特殊情况下可以降低服务器等级，甚至关闭。



微服务------鉴权&授权：token



微服务------瞬态故障处理：

Polly是一种.net弹性和瞬态处理库，允许我们以非常顺畅和线程安全的方式来执行诸如 重试，断路，超时，故障恢复等策略。

微服务------分布式追踪(常用软件Skyworking)：

分布式追踪和APM的Server端，它将包含Collector,Stroage,独立的WebUI,并使用OpenTracing规范来设计追踪数据。

微服务------ExceptionLess：

ExceptionLess:开源的日志收集和分析框架，能为应用程序提供实时错误，特性和日志报告。

微服务------统一配置中心：

Apollo:配置管理平台，能够集中化管理应用不同环境，不同集群的配置，配置修改后能实时推送到应用端，并且具备规范的权限、流程治理等特性。

微服务------分布式事务：

CAP理论

微服务------容器化docker：

Docker:是一个开源的应用程序引擎，可以打包应用以及依赖包到一个可移植的镜像中，然后发布到任何流行的linux或者windows机器上，也可以实现虚拟化

微服务------容器编排K8S（管理docker）:

Kubernetes编排容器，是管理应用的全生命周期的工具，从创建应用/部署，应用提供服务，扩容缩容，更新，都非常方便，而且可以做到故障自愈。

微服务------CI/CD：

Jenkins是一个开源的，提供友好操作界面的持续集成(CI)工具，主要用于持续、自动的构建/测试软件项目、监控外部任务的运行。