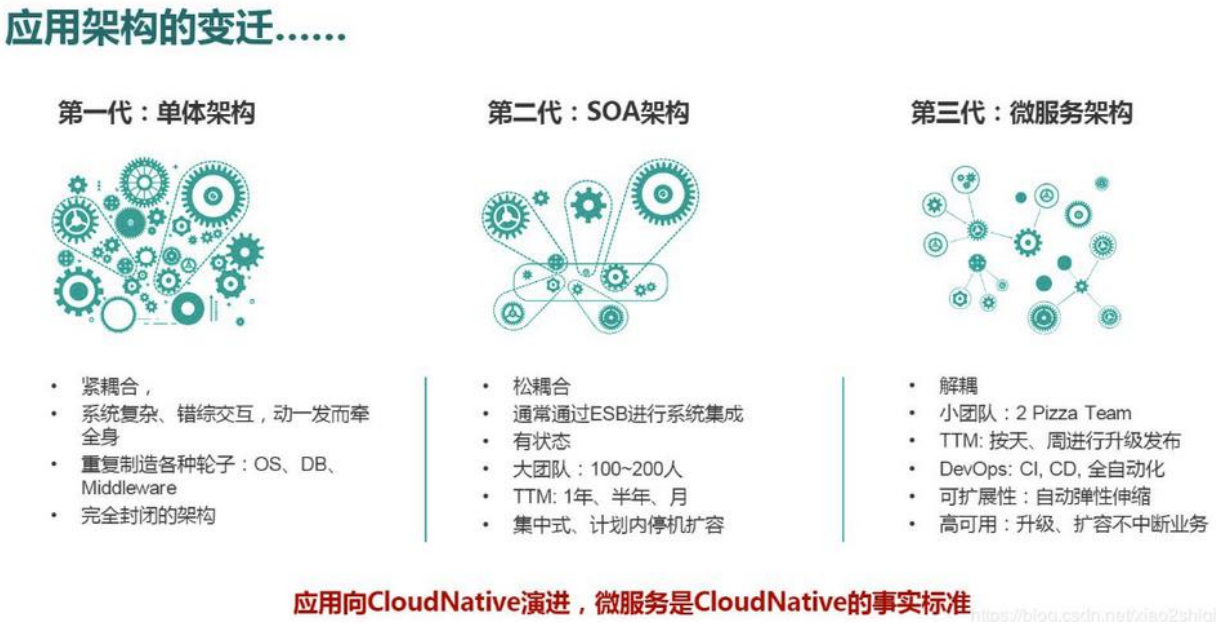
什么是微服务?

简单举例：一艘航空母舰作战能力虽然很强，但是弱点太明显，就是防御能力太差，单艘的航空母舰很少单独行动，通常航空母舰战斗群才是主要军事力量，你可以把单艘航母理解为的单体应用（防御差，机动性不好），把航母战斗群（调度复杂，维护费用高）理解为微服务。

大部分的开发者经历和开发过单体应用，无论是传统的 Servlet + JSP，还是 SSM，还是现在的 SpringBoot，它们都是单体应用，那么长期陪伴我们的单体应用有什么弊端？我们是面临了什么问题，导致我们要抛弃单体应用转向微服务架构?

- 部署成本高（无论是修改1行代码，还是10行代码，都要全量替换）
- 改动影响大，风险高（不论代码改动多小，成本都相同）
- 因为成本高，风险高，所以导致部署频率低（无法快速交付客户需求）



微服务的阶段

前期阶段，设计阶段，技术阶段

前期阶段，大致要做好如下事情:

- 和多方充分沟通，确保能符合客户和组织的需求，并且得到认同
- 和团队沟通，让队友（开发/测试/运维）理解，并且积极投入
- 和业务部门沟通，指定版本计划和上线时间

设计阶段，参考 Sam Newman 的著作《微服务设计》，单微服务必须要满足以下的条件，才符合微服务的基本要求:

- 标准的 REST 风格接口（基于 HTTP 和 JSON 格式）
- 独立部署，避免共享数据库（避免因为数据库而影响整个分布式系统）

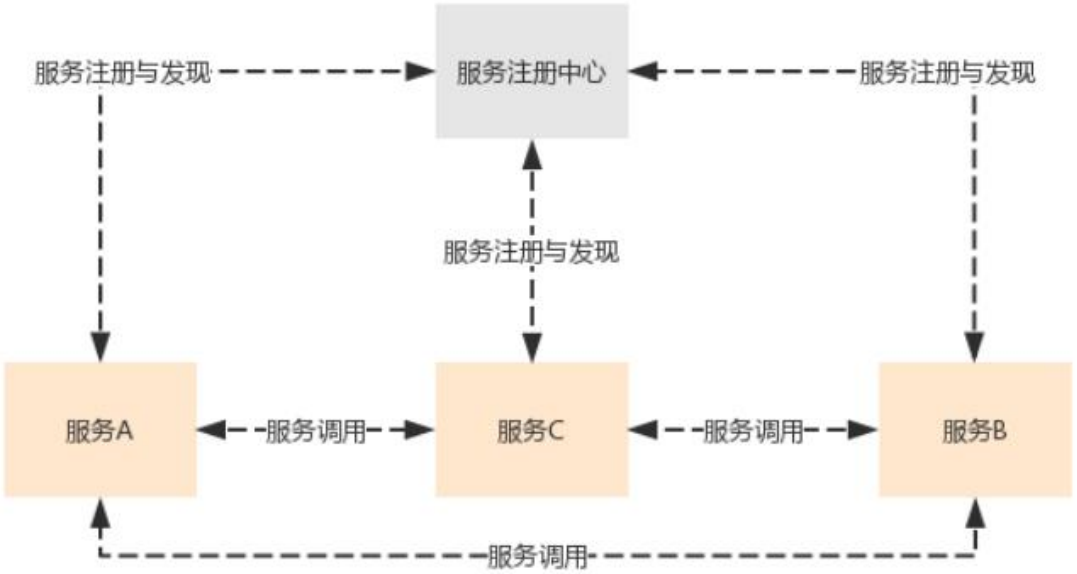
- 业务上的高内聚，减少依赖（从设计上要避免服务过大或者太小）

庞大的分布式系统，需要强大基础设施来支撑，微服务涉及哪些基础设施？

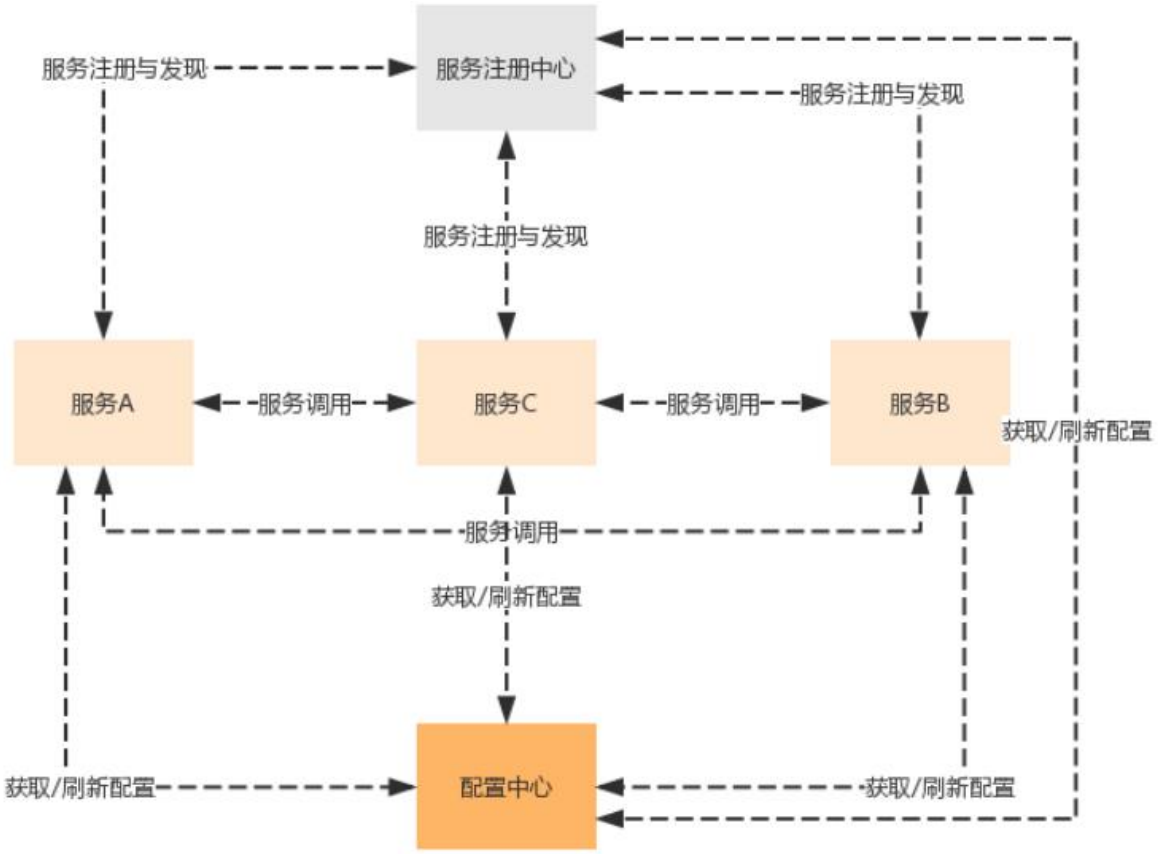
- CI/CD和自动化（分布式系统几乎不可能通过人工手动发布）
- 虚拟化技术（要保证微服务运行环境隔离，目前行业主流的是使用 Docker 容器）
- 日志聚合，全链路监控（高度可观察和分析诊断问题）

微服务典型架构

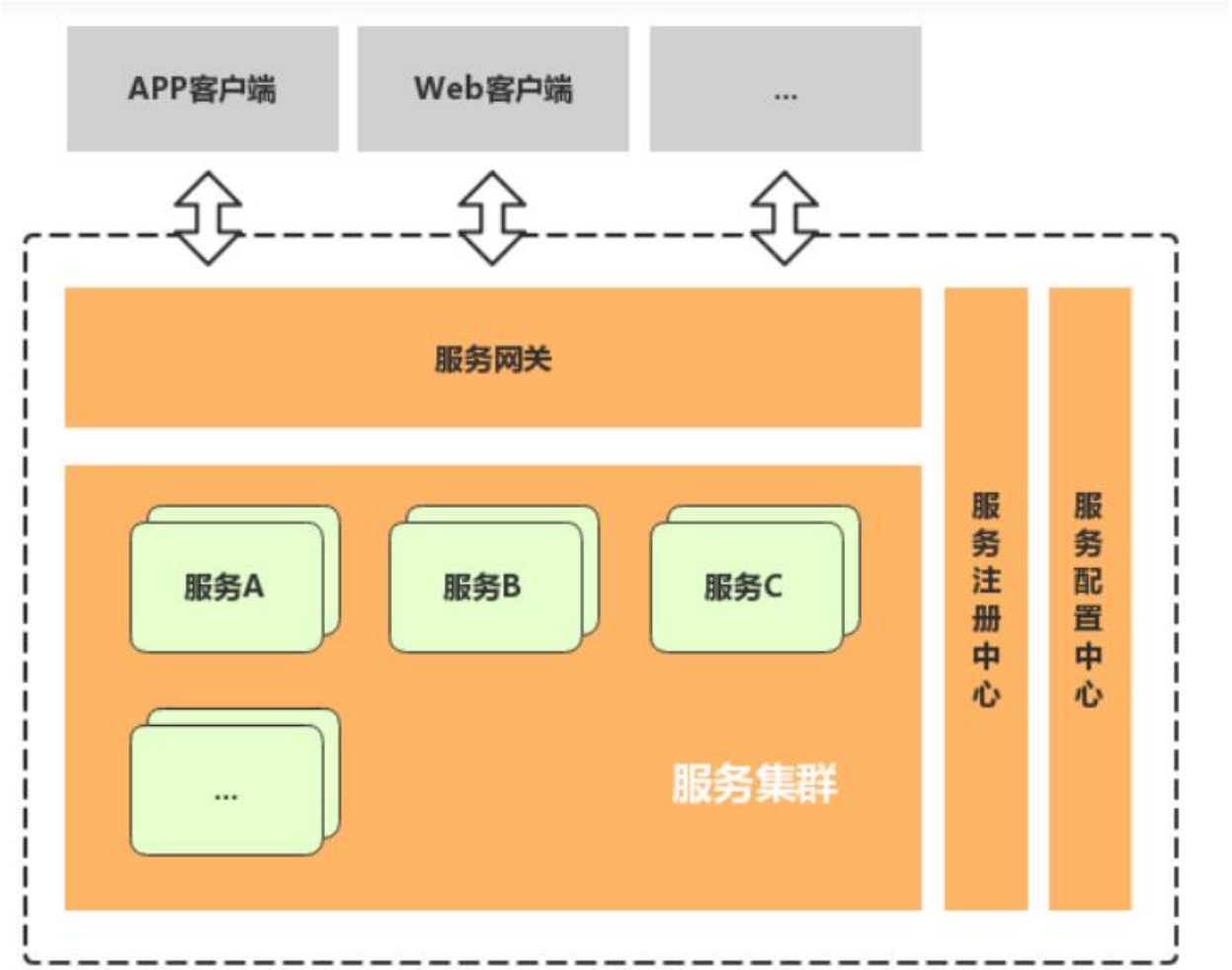
微服务架构，核心是为了解决应用微服务化之后的服务治理问题。
应用微服务化之后，首先遇到的第一个问题就是服务发现问题，一个微服务如何发现其他微服务呢？最简单的方式就是每个微服务里面配置其他微服务的地址，但是当微服务数量众多的时候，这样做明显不现实。所以需要使用到微服务架构中的一个最重要的组件：服务注册中心，所有服务都注册到服务注册中心，同时也可以从服务注册中心获取当前可用的服务清单：



解决服务发现问题后，接着需要解决微服务分布式部署带来的第二个问题：服务配置管理的问题。当服务数量超过一定程度之后，如果需要在每个服务里面分别维护每一个服务的配置文件，运维人员估计要哭了。那么，就需要用到微服务架构里面第二个重要的组件：**配置中心**，微服务架构就变成下面这样了：



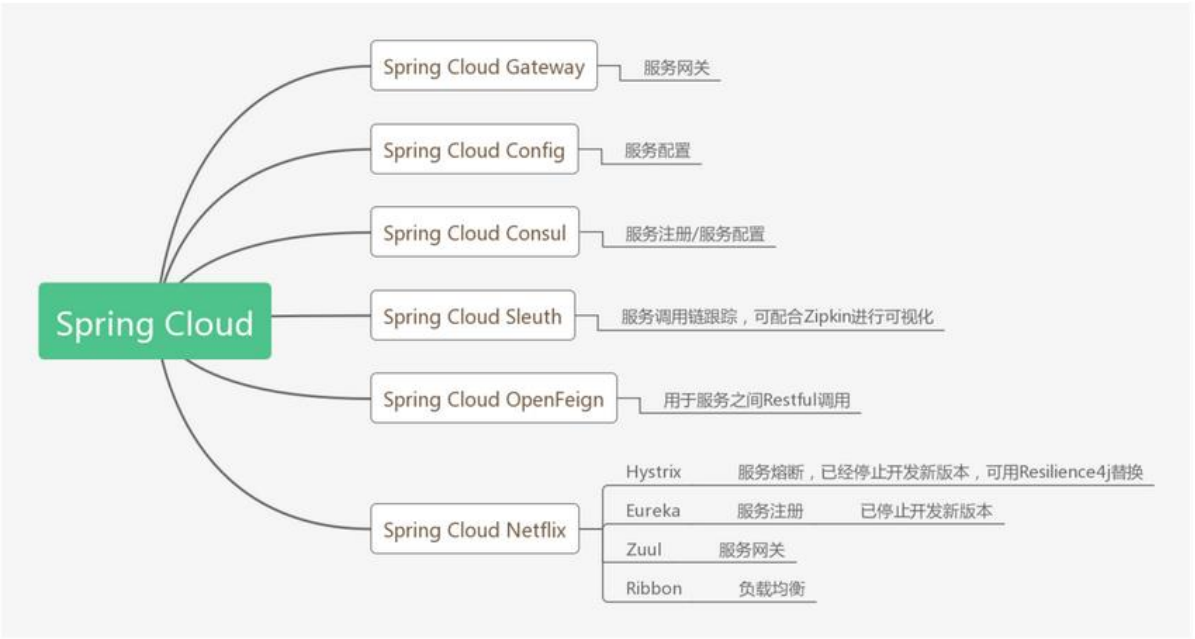
以上应用内部的服务治理，当客户端或外部应用调用服务的时候怎么处理呢？服务A可能有多个节点，服务A、服务B和服务C的服务地址都不同，服务授权验证在哪里做？这时，就需要使用到服务网关提供统一的服务入口，最终形成典型微服务架构：



微服务框架

目前国内企业使用的微服务框架主要是Spring Cloud和Dubbo（或者DubboX），但是Dubbo那两年的停更严重打击了开发人员对它的信心，Spring Cloud已经逐渐成为主流，比较两个框架的优劣势的文章在网上有很多，这里就不重复了，选择什么框架还是按业务需求来吧，业务框架决定技术框架。

Spring Cloud全家桶提供了各种各样的组件，基本可以覆盖微服务的服务治理的方方面面，以下列出了Spring Cloud一些常用组件：



具体搭建还没有摸索透，仅仅收集总结了一下简单的概念，未完待续。