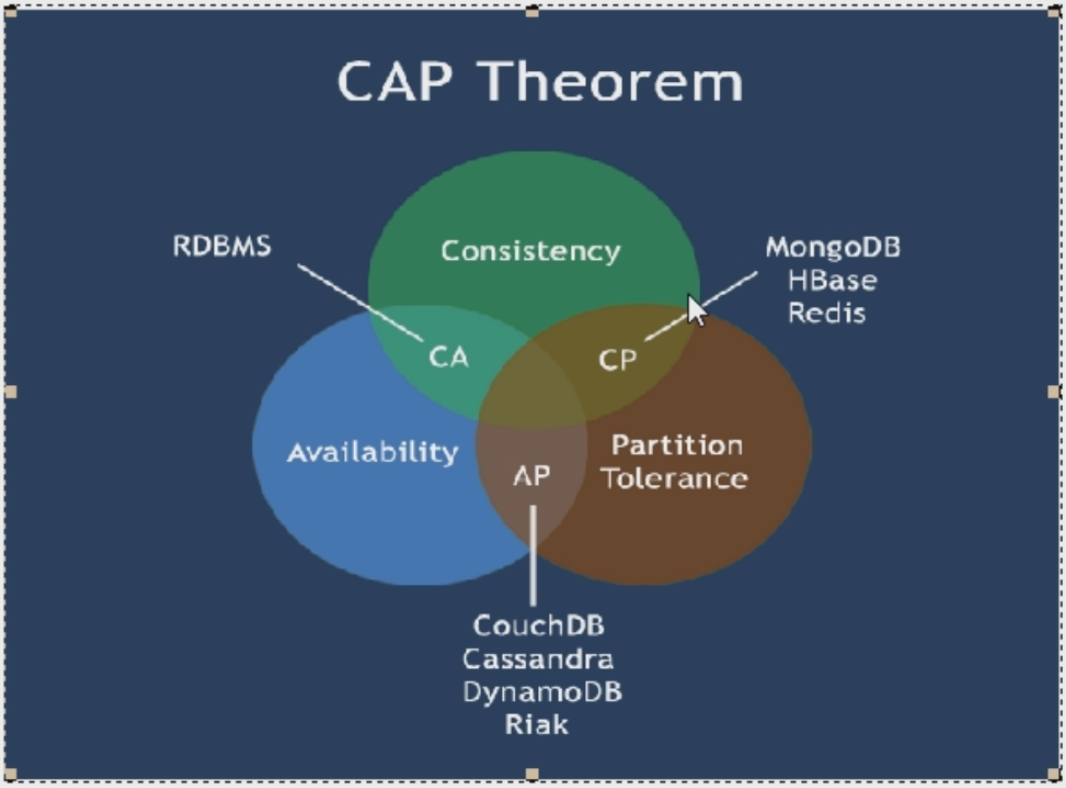
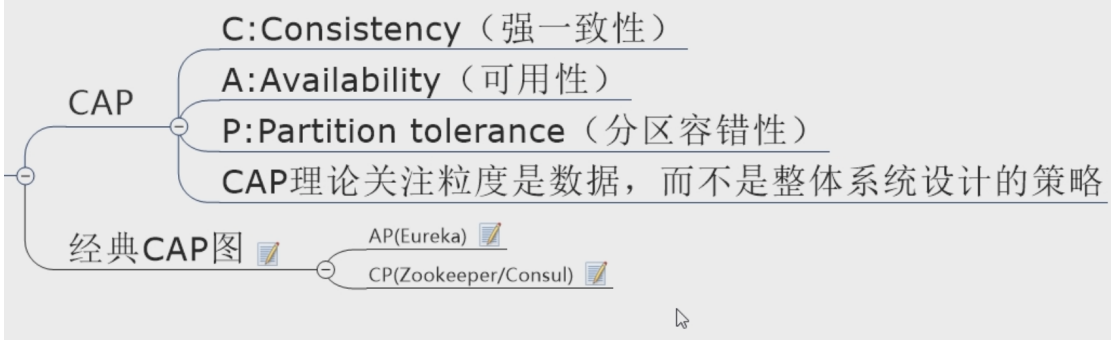
CAP理念

组件名	语言	CAP	服务健康检查	对外暴露接口	Spring Cloud集成
Eureka	Java	AP	可配支持	HTTP	已集成
Consul	Go	CP	支持	HTTP/DNS	已集成
Zookeeper	Java	CP	支持	客户端	已集成





最多只能同时较好的满足两个。

CAP理论的核心是：一个分布式系统不可能同时很好的满足一致性，可用性和分区容错性这三个需求，
因此，根据 CAP 原理将 NoSQL 数据库分成了满足 CA 原则、满足 CP 原则和满足 AP 原则三 大类：
CA - 单点集群，满足一致性，可用性的系统，通常在可扩展性上不太强大。
CP - 满足一致性，分区容忍必的系统，通常性能不是特别高。
AP - 满足可用性，分区容忍性的系统，通常可能对一致性要求低一些。

服务注册中心解决的问题

- ```
  # 服务注册中心解决的问题

      服务注册中心主要解决两个关键问题：服务注册和服务发现。

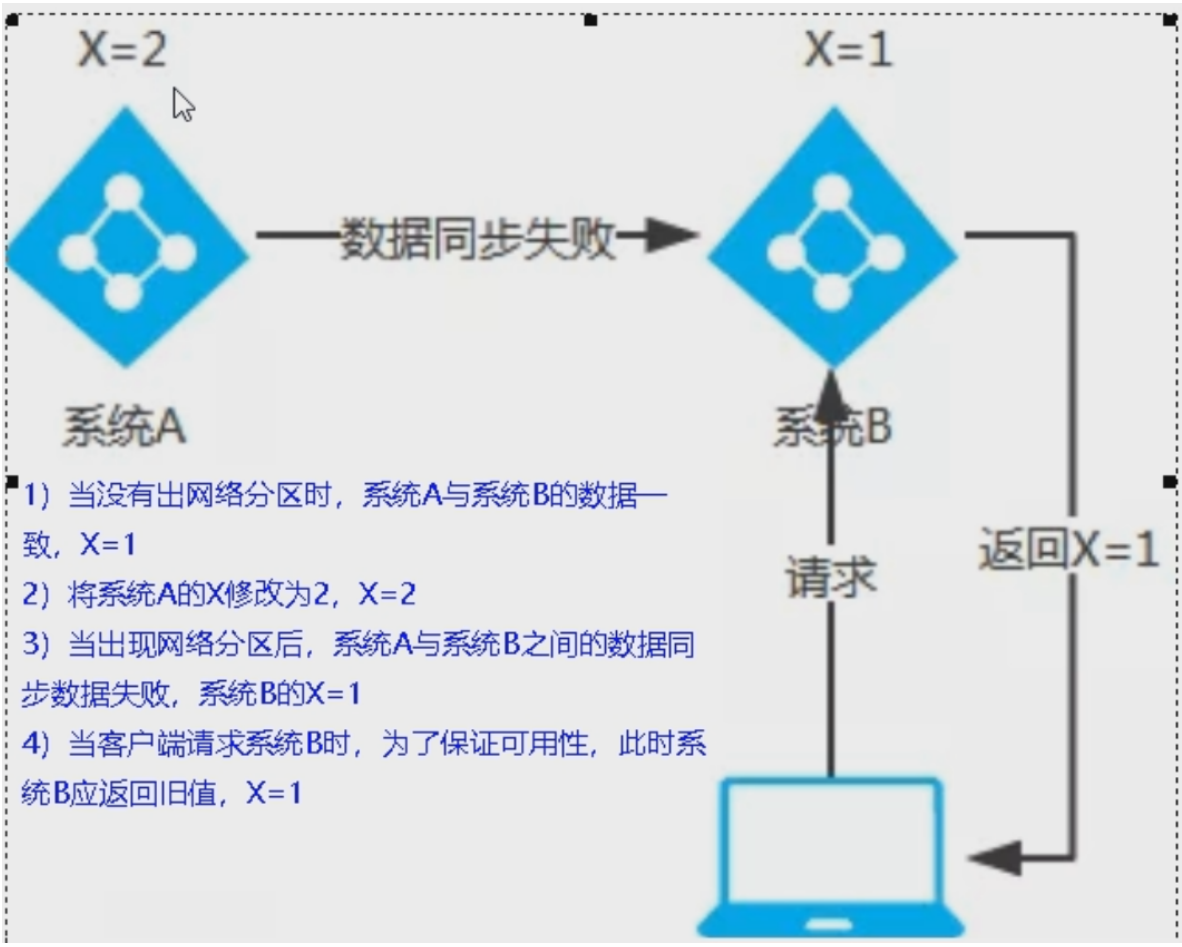
  - 服务注册：服务实例将自身服务信息注册到注册中心。这部分服务信息包括服务所在主机IP和提供服务的Port，以及暴露服务自身状态以及访问协议等信息。
  - 服务发现：服务实例请求注册中心获取所依赖服务信息。服务实例通过注册中心，获取到注册到其中的服务实例的信息，通过这些信息去请求它们提供的服务。

      除了这两个核心功能之外，一般服务注册中心还需要监控服务实例的运行状态，负载均衡等问题。

  - 监控：服务实例一直处于动态的变化中，因此我们需要监控服务实例的健康状况，从注册中心剔除无用的服务。一般实现心跳连接等。
  - 负载均衡：在一个服务有多个实例的情况下，我们需要根据负载均衡策略正确处理请求。
  ```

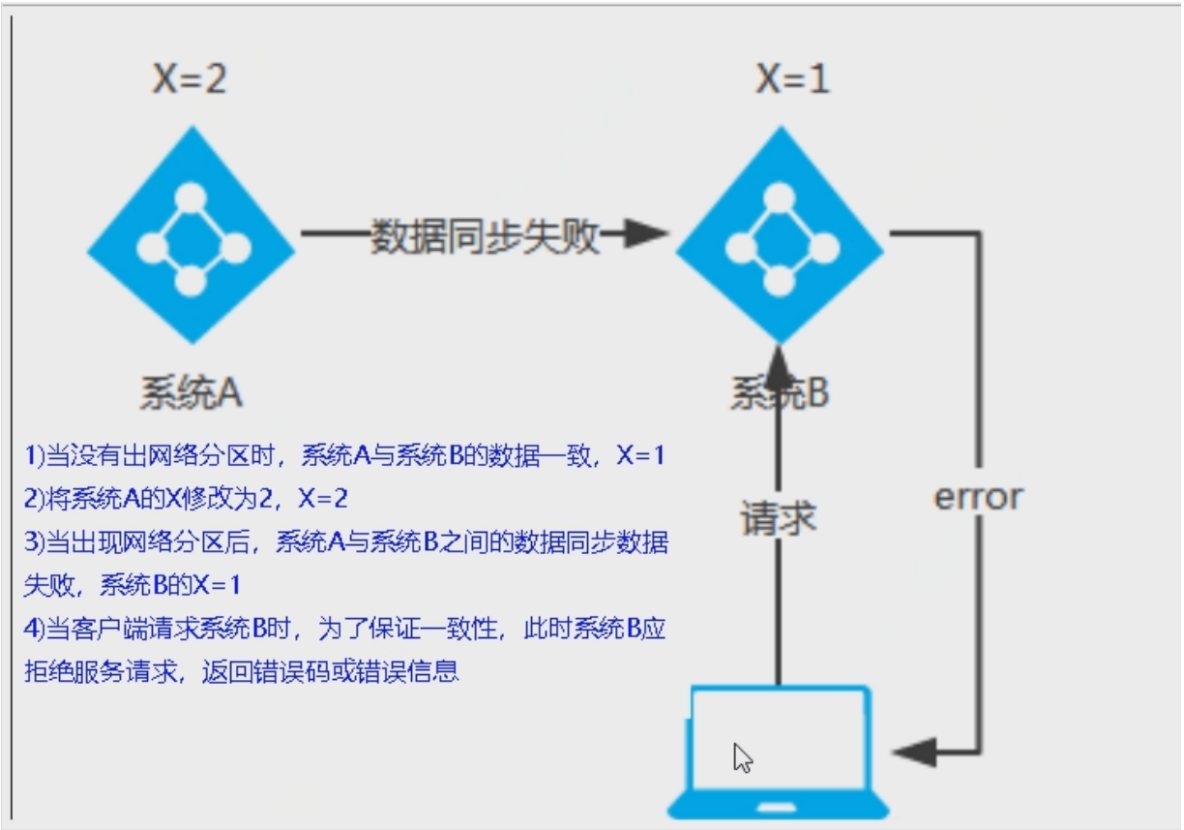
AP架构 （eureka）

AP架构
当网络分区出现后，为了保证可用性，系统B可以返回旧值，保证系统的可用性。
结论：违背了一致性C的要求，只满足可用性和分区容错，即AP
  ```



CP架构（ZK/consul）

> CP架构
> 当网络分区出现后，为了保证一致性，就必须拒接请求，否则无法保证一致性
> 结论：违背了可用性A的要求，只满足一致性和分区容错，即CP



主流注册中心产品

	Nacos	Eureka	Consul	CoreDNS	Zookeeper
一致性协议	CP+AP	AP	CP	—	CP
健康检查	TCP/HTTP/MYSQL/Client Beat	Client Beat	TCP/HTTP/gRPC/Cmd	—	Keep Alive
负载均衡策略	权重/ metadata/Selector	Ribbon	Fabio	RoundRobin	—
雪崩保护	有	有	无	无	无
自动注销实例	支持	支持	不支持	不支持	支持
访问协议	HTTP/DNS	HTTP	HTTP/DNS	DNS	TCP
监听支持	支持	支持	支持	不支持	支持
多数据中心	支持	支持	支持	不支持	不支持
跨注册中心同步	支持	不支持	支持	不支持	不支持
SpringCloud集成	支持	支持	支持	不支持	不支持
Dubbo集成	支持	不支持	不支持	不支持	支持
K8S集成	支持	不支持	支持	支持	不支持