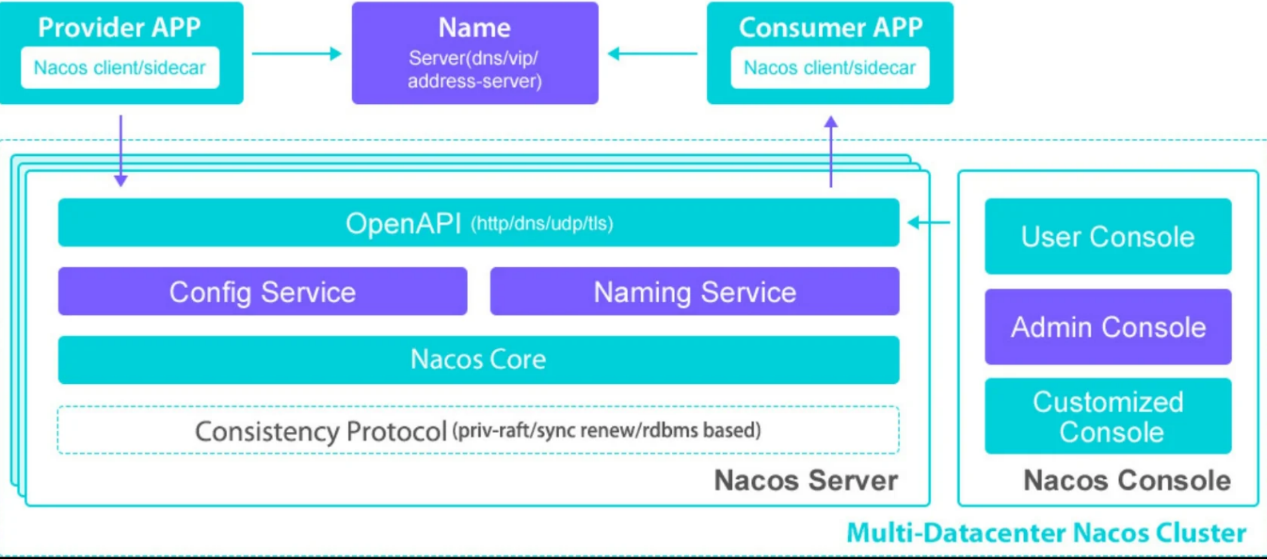
Nacos架构图：



Provider app：服务提供者

Consumer app：服务消费者

Name server：通过VIP（vritual IP）或者DNS的方式实现nacos高可用集群的服务路由

Nacos server：nacos服务提供者，里面包含的open api是功能访问的入口，config service、naming service 是nacos提供的配置服务，名字服务模块。Consistency protocol是一致性协议，用来实现nacos集群节点的数据同步，这里使用的是raft算法（使用类似算法的中间件还有Etcd、Redis烧饼选举）

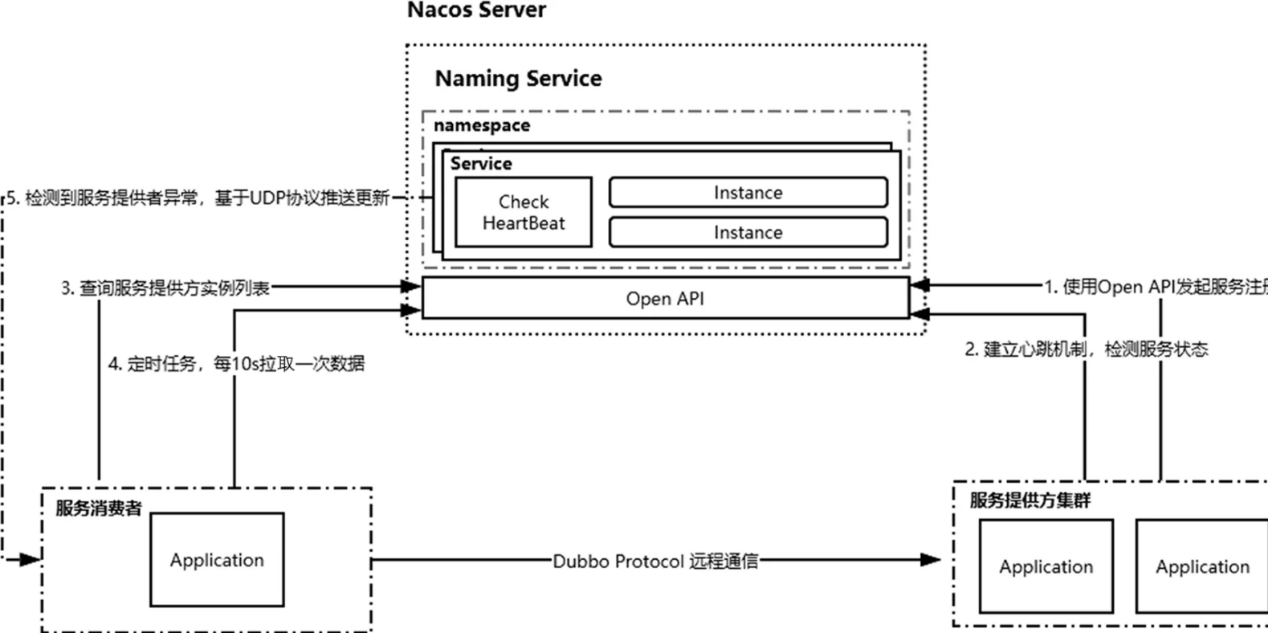
Nacos console：nacos控制台

注册中心原理

服务实例在启动时注册到服务注册表，并在关闭时注销。

服务消费者查询服务注册表，获得可用实例

服务注册中心需要调用服务实例的健康检查api来验证它是否能够处理请求。



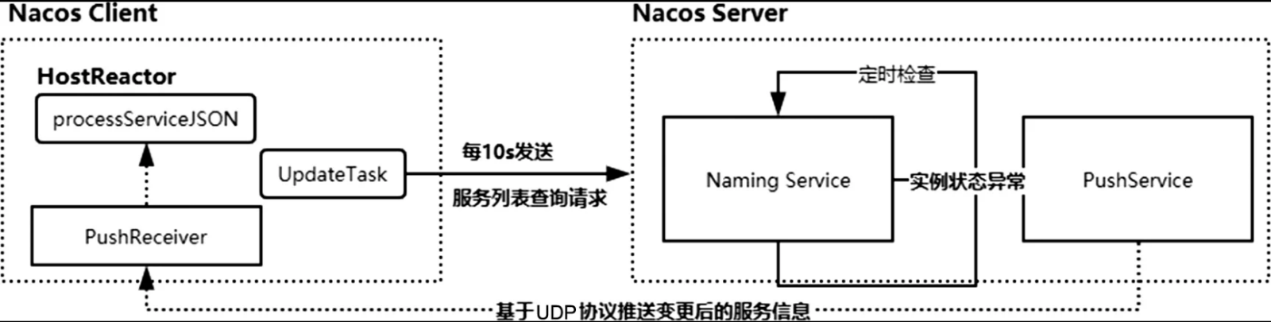
服务动态感知：

Nacos客户端有一个HostReactor类，他的功能是实现服务的动态更新，基本原理是：

客户端发起事件订阅后，在HostReactor中有一个UpdateTask线程，每10s发送一次Pull请求，获得服务端最新的地址列表。

对于服务端，它和服务提供者的实例之间维持了心跳检测，一旦服务提供者出现异常。则会发送一个push消息给nacos客户端，也就是服务消费者。

服务消费者收到请求之后，使用hostReactor中提供的processServiceJSON解析消息，并更新本地服务地址列表



Nacos采用的是pull模式，但并不是简单的pull，而是一种长轮询机制，他结合push和pull两者的优势。客户端采用长轮询的方式定时发起pull请求，去检查服务端配置信息是否发生了变更，如果发生了变更，则客户端会根据变更的数据获取最新的配置。所谓长轮询，是客户局端发起轮询请求之后，服务端如果有配置发生变更，就直接返回



如果客户端发起pull请求后，发现服务端的配置和客户端的配置保持一致，那么服务端会先hold住这个请求，也就是服务端拿到这个连接之后再指定的时间段内一直不返回结果，直到这段时间内配置发生改变，服务端会把原先hold住的请求进行返回。

Nacos服务端收到请求之后，先检查配置是否发生变更，如果没有，则设置一个定时任务，延期29.5s执行，并且把当前客户端长轮询加入allSubs队列，这时候有两种方式触发该连接结果的返回：

1. 在等待29.5s后触发自动检查机制，这个时候不管配置有没有发生变化，都会把结果返回客户端。而29.5s就是这个长连接保持的时间。
2. 在29.5s内任意一个时刻，通过nacosDashboard或者api的方式对配置进行了修改，这会触发一个事件机制，监听到该事件的任务会便利allSubs队列，找到发生变更的配置项对应的clientLongPolling任务，将变更的数据通过该任务中的链接进行返回，就完成了一次“推送”操作

这样既能保证客户端实时感知配置的变化，也降低服务端的压力。其中，这个长连接的会话超时时间默认是30s

