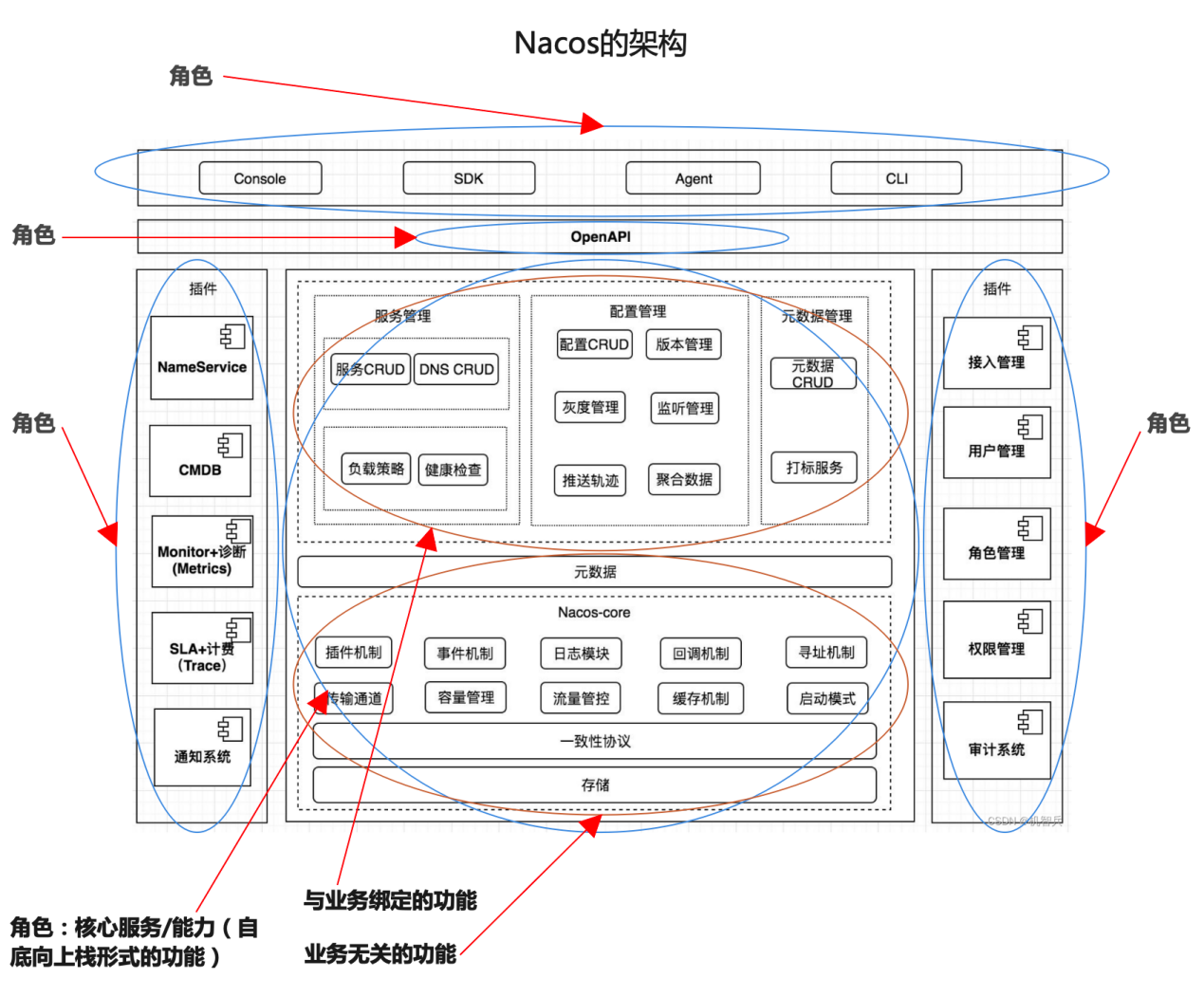
Nacos的整体架构分为用户层、业务层、内核层和插件，用户层主要解决用户使用的易用性问题，业务层主要解决服务发现和配置管理的功能问题，内核层解决分布式系统⼀致性、存储、高可用等核心问题，插件解决扩展性问题。



1、用户层

* OpenAPI：暴露标准 Rest 风格 HTTP 接口，简单易用，方便多语言集成。
* Console：易用控制台，做服务管理、配置管理等操作。
* SDK：多语言 SDK，目前几乎支持所有主流编程语言。
* Agent：Sidecar 模式运行，通过标准 DNS 协议与业务解耦。
* CLI：命令行对产品进行轻量化管理，像 git ⼀样好用。

2、业务层

* 服务管理：实现服务 CRUD，域名 CRUD，服务健康状态检查，服务权重管理等功能。
* 配置管理：实现配置管 CRUD，版本管理，灰度管理，监听管理，推送轨迹，聚合数据等功能。
* 元数据管理：提供元数据 CURD 和打标能力，为实现上层流量和服务灰度非常关键。

3、内核层

* 插件机制：实现三个模块可分可合能力，实现扩展点 SPI 机制，用于扩展自己公司定制。
* 事件机制：实现异步化事件通知，SDK 数据变化异步通知等逻辑，是 Nacos 高性能的关键部分。
* 日志模块：管理日志分类，日志级别，日志可移植性（尤其避免冲突），日志格式，异常码+帮助文档。
* 回调机制：SDK 通知数据，通过统⼀的模式回调用户处理。接口和数据结构需要具备可扩展性。
* 寻址模式：解决 Server IP 直连，域名访问，Nameserver 寻址、广播等多种寻址模式，需要可扩展。
* 推送通道：解决 Server 与存储、Server 间、Server 与 SDK 间高效通信问题。
* 容量管理：管理每个租户，分组下的容量，防止存储被写爆，影响服务可用性。
* 流量管理：按照租户，分组等多个维度对请求频率，长链接个数，报文大小，请求流控进行控制。
* 缓存机制：容灾目录，本地缓存，Server 缓存机制，是 Nacos 高可用的关键。
* 启动模式：按照单机模式，配置模式，服务模式，DNS 模式模式，启动不同的模块。
* ⼀致性协议：解决不同数据，不同⼀致性要求情况下，不同⼀致性要求，是 Nacos 做到 AP 协议的关键。
* 存储模块：解决数据持久化、非持久化存储，解决数据分片问题。

4、插件

* Nameserver：解决 Namespace 到 ClusterID 的路由问题，解决用户环境与 Nacos 物理环境映射问题。
* CMDB：解决元数据存储，与三方 CMDB 系统对接问题，解决应用，人，资源关系。
* Metrics：暴露标准 Metrics 数据，方便与三方监控系统打通。
* Trace：暴露标准 Trace，方便与 SLA 系统打通，日志白平化，推送轨迹等能力， 并且可以和计量计费系统打通。
* 接入管理：相当于阿里云开通服务，分配身份、容量、权限过程。
* 用户管理：解决用户管理，登录，SSO 等问题。
* 权限管理：解决身份识别，访问控制，角色管理等问题。
* 审计系统：扩展接口方便与不同公司审计系统打通。
* 通知系统：核心数据变更，或者操作，方便通过。