用Kong构建混合环境下的服务网络

蔡书

独立顾问/PolarisTech联合创始人

极客时间 | 企业服务

这是一封正经“福利”贴

想做团队的领跑者 需要迈过这些“槛”

成长型企业，易忽视人才体系化培养 企业转型加快，团队能力又跟不上	VS	从基础到进阶，超100+一线实战 技术专家带你系统化学习成长
团队成员技能水平不一， 难以一“敌”百人需求	VS	解决从小白到资深技术人所遇到 80%的问题
寻求外部培训，奈何价更高且 集中式学习	VS	多样、灵活的学习方式，包括 音频、图文 和视频
学习效果难以统计，产生不良循环	VS	获取员工学习报告，查看学习 进度，形成闭环

课程顾问「橘子」

回复「QCon」
免费获取
学习解决方案

 # 极客时间企业账号 # 解决技术人成长路上的学习问题

自我介绍

蔡书，独立顾问，PolarisTech联合创始人；开源软件和解决方案的爱好者，技术上崇尚简单实用，追求稳定、简洁、高效的解决方案。

目录

- 分布式环境与服务网络
- Kong——Nginx家族小伙伴
- 基础的基础——LBaaS
- 南北向与东西向流量
- 数据平面：sidecar与代理
- 控制平面：多租户、配置和缓存
- 部署环境：物理机、云、虚拟化、容器
- 跨微服务平台
- 监控与服务质量

引言

随着微服务架构的流行，服务之间流量的管理变成了新的问题和话题。为了解决这类问题，以Istio为代表的开源软件提出了service mesh的概念，第一次把这个领域的问题、功能、解决办法做了汇总，完成了这个领域概念的标准化和体系化的工作。目前大多数的service mesh解决的问题都是依赖或者绑定到微服务平台和框架上（比如Istio构建在K8s之上）。但是现实当中，使用环境往往是复杂的，尤其是企业环境，一般都包含多个版本、多种微服务平台（比如K8s，Dubbo，Spring Cloud等）；同时部署环境也涵盖了公有云、虚拟机、物理机、容器等；在地域上也有着跨数据中心的需求。一起了解如何基于Kong来解决这些问题。

分布式环境与服务网络

现状：

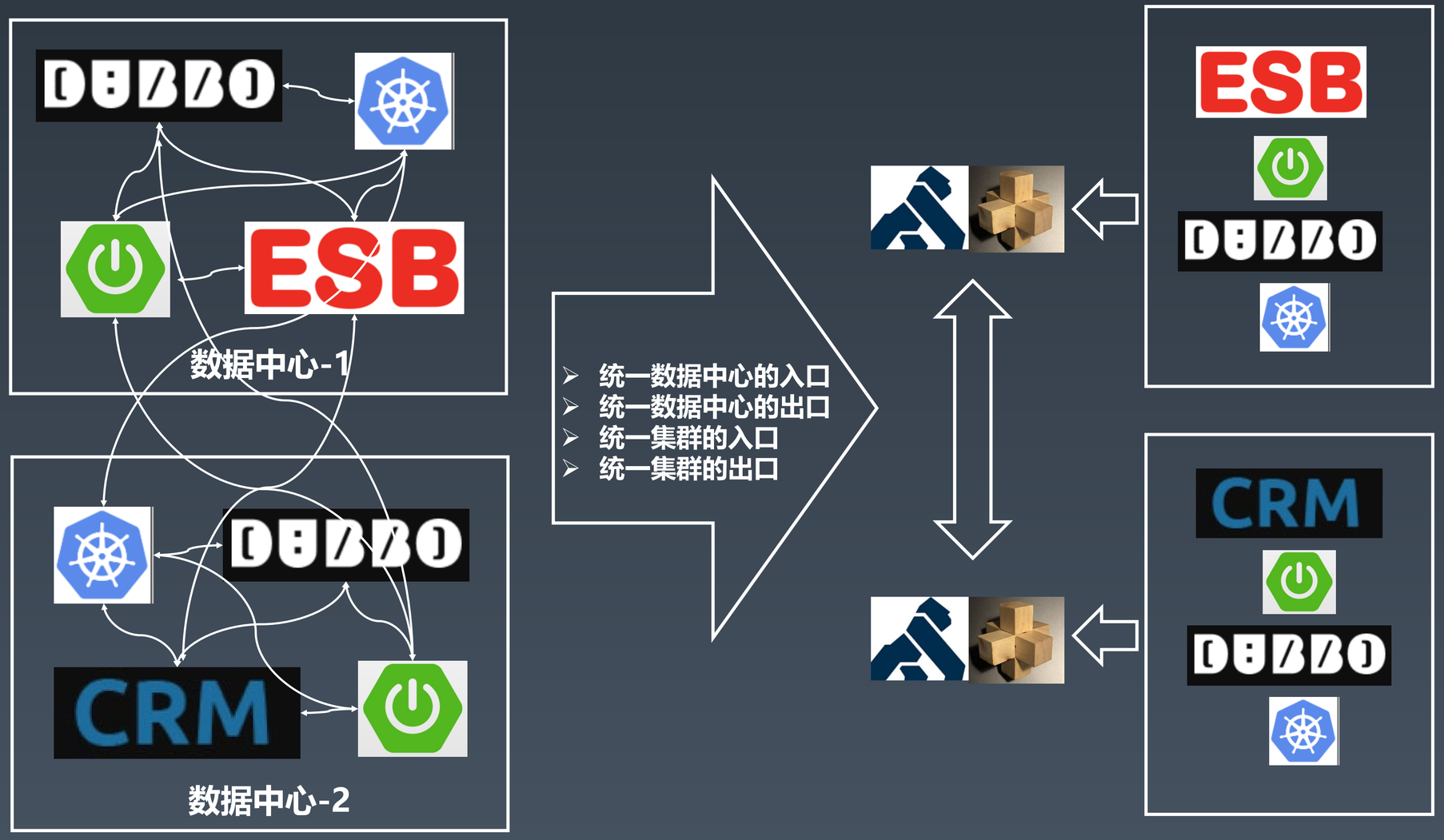
- 分布式从一种架构选择，变成一种客观存在
- 多平台：Spring，Dubbo，K8S
- 多堆栈：Java，Node.js，PHP，Python ...
- 多中心：两地三中心
- 传统系统： Core Banking、ESB、CRM、ERP

问题：

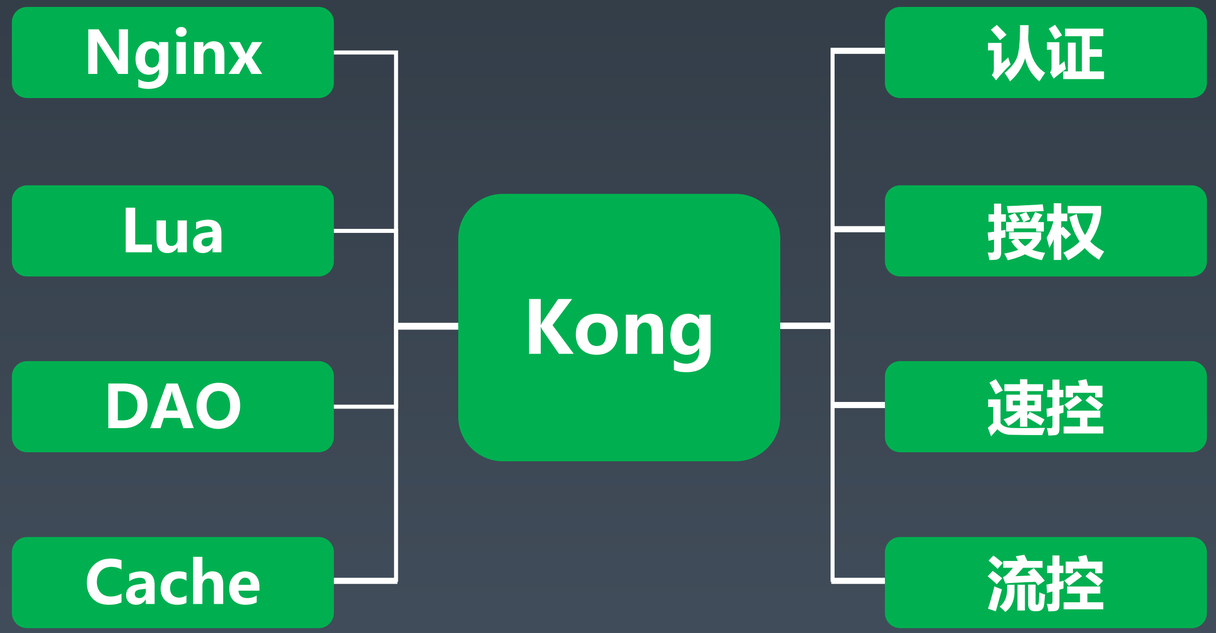
- 开通全链路访问是一个繁琐、容易出错的过程
- 访问规则的管理复杂、冗余、过期配置大量存在
- 防火墙无法有效实现应用层访问控制
- 缺乏应用层的统一视图
- 跨集群的服务间依赖关系管理复杂

“随着Service Mesh概念的提出和实践，分布式架构从“功能导向”向“服务质量导向”演化。”

化繁为简，分而治之


DUBBO
ESB
数据中心-1
CRM
DUBBO
数据中心-2
统一数据中心的入口
统一数据中心的出口
统一集群的入口
统一集群的出口
ESB
DUBBO
CRM
DUBBO
应用层解决
框架层解决
协议层解决
更通用，更兼容

Kong——Nginx家族小伙伴



Nginx家族：

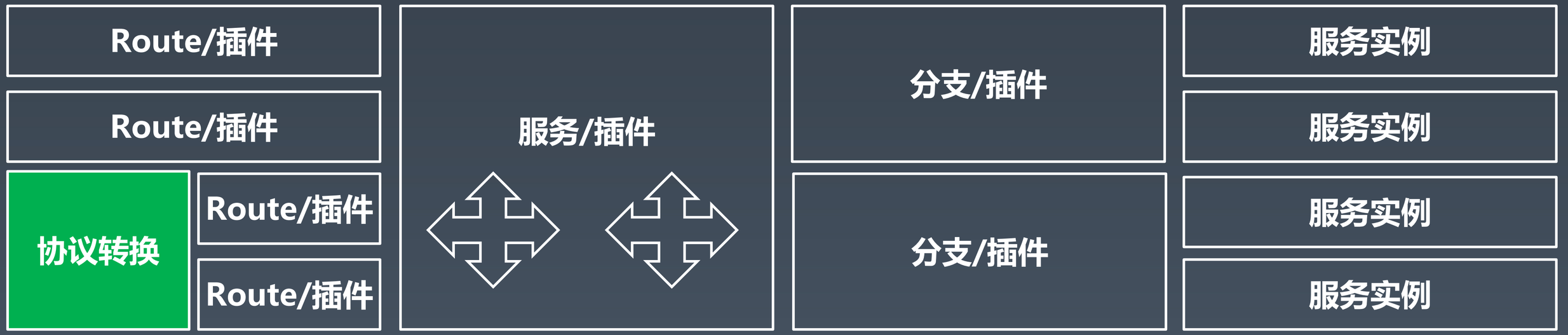
- Nginx -> OpenResty -> Kong
- Lua开发，插件体系
- 采用数据库维护配置一致
- 高效、高速、高稳定
- GitHub 20K+ star

软负载即服务

- 优良的LB是流量管理的基础
- 高吞吐、高可靠
- 扩容——水平扩展
- 便捷的部署
- REST 接口，基础设施可编程
- 配置同步
- 与已有体系集成——监控、日志、跟踪
- 功能可扩展

Tips:

- HTTP为REQ/RES提供了标准模型
- 充分利用HTTP Keep-Alive
- 根据网络条件使用HTTP短连接



南北向和东西向流量

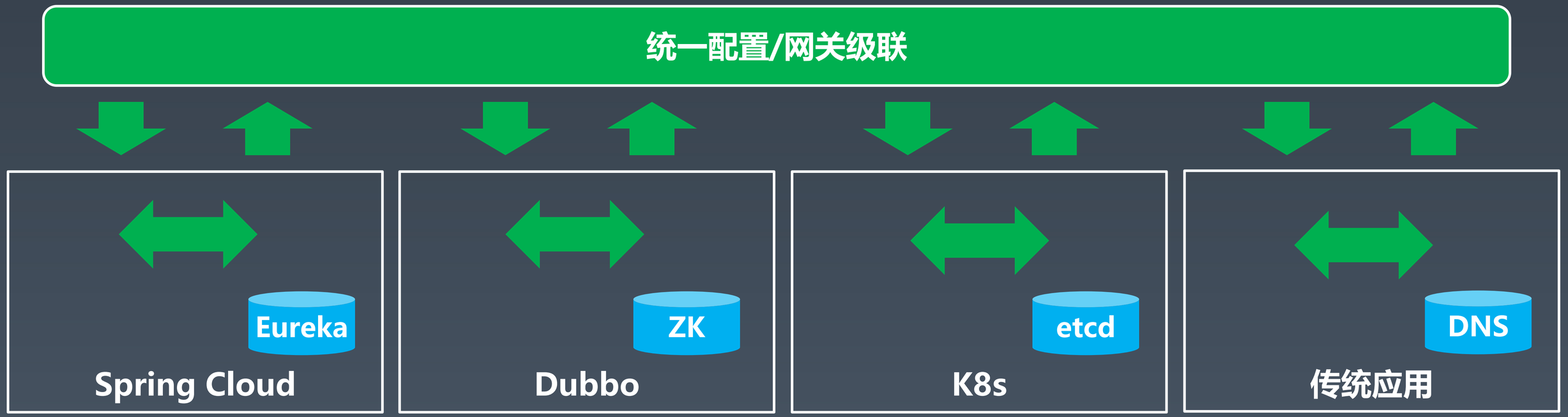
南北流量：
- Ingress
- Egress

东西流量：
- 服务调用
- 条件路由
- 全局路由

服务管理：
- 注册
- 发现
- 发布(蓝绿/滚动)

断路保护：
- 熔断
- 降级
- 限速

Tips:
- ✓ 通过请求信息匹配规则
- ✓ 匹配和规则区分开
- ✓ 复用东西/南北通用规则



数据平面：Sidecar与代理

Sidecar(istio)：

- 两次代理
- Iptables拦截
- 容器?
- Linux?

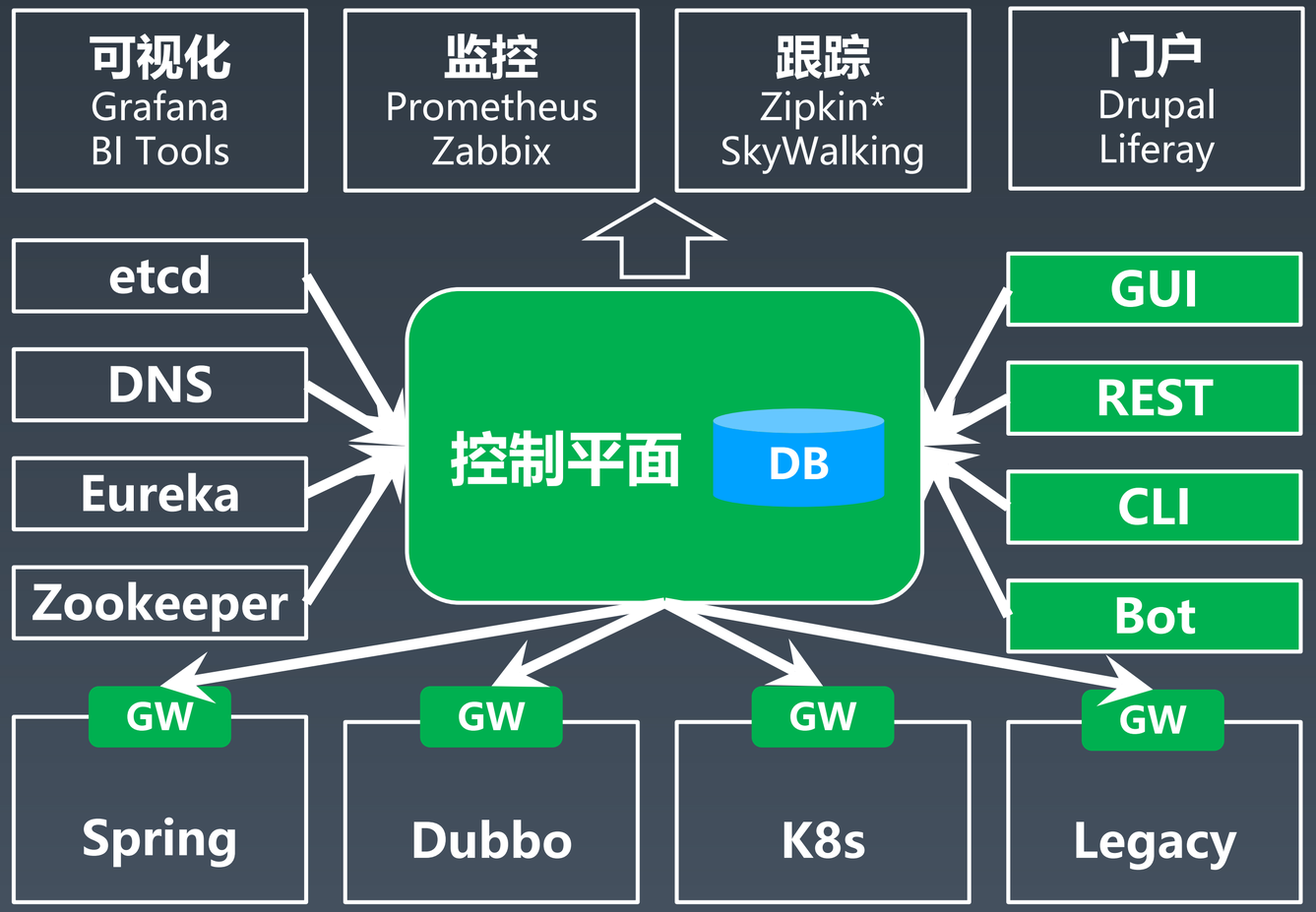
代理：

- HTTP
- Socks5
- 物理机/容器/虚拟机
- Linux/Windows

Tips：

- ✓ 环境变量可以设置代理
- ✓ Socks5支持域名和认证

控制平面：多租户、配置和缓存



- 全局化配置：不再“铁路警察各管一段”
- 多租户与自服务：尽量授权给服务维护者做配置变更
- 自动化：从服务配置中心获取配置
- 可视化：操作可视化，结果可视化
- 流程：加入必要的审批和review
- 版本化：配置的回溯能力

Tips:

- ✓ 配置用数据库
- ✓ 执行时用缓存
- ✓ 统一建模

部署环境：物理机、云、虚拟化、容器

部署方式：

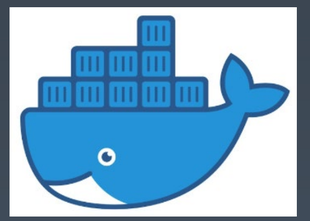
- yum install … ; systemctl start…
- docker run …
- kubectl -f …
- ansible-playbook …

好处：

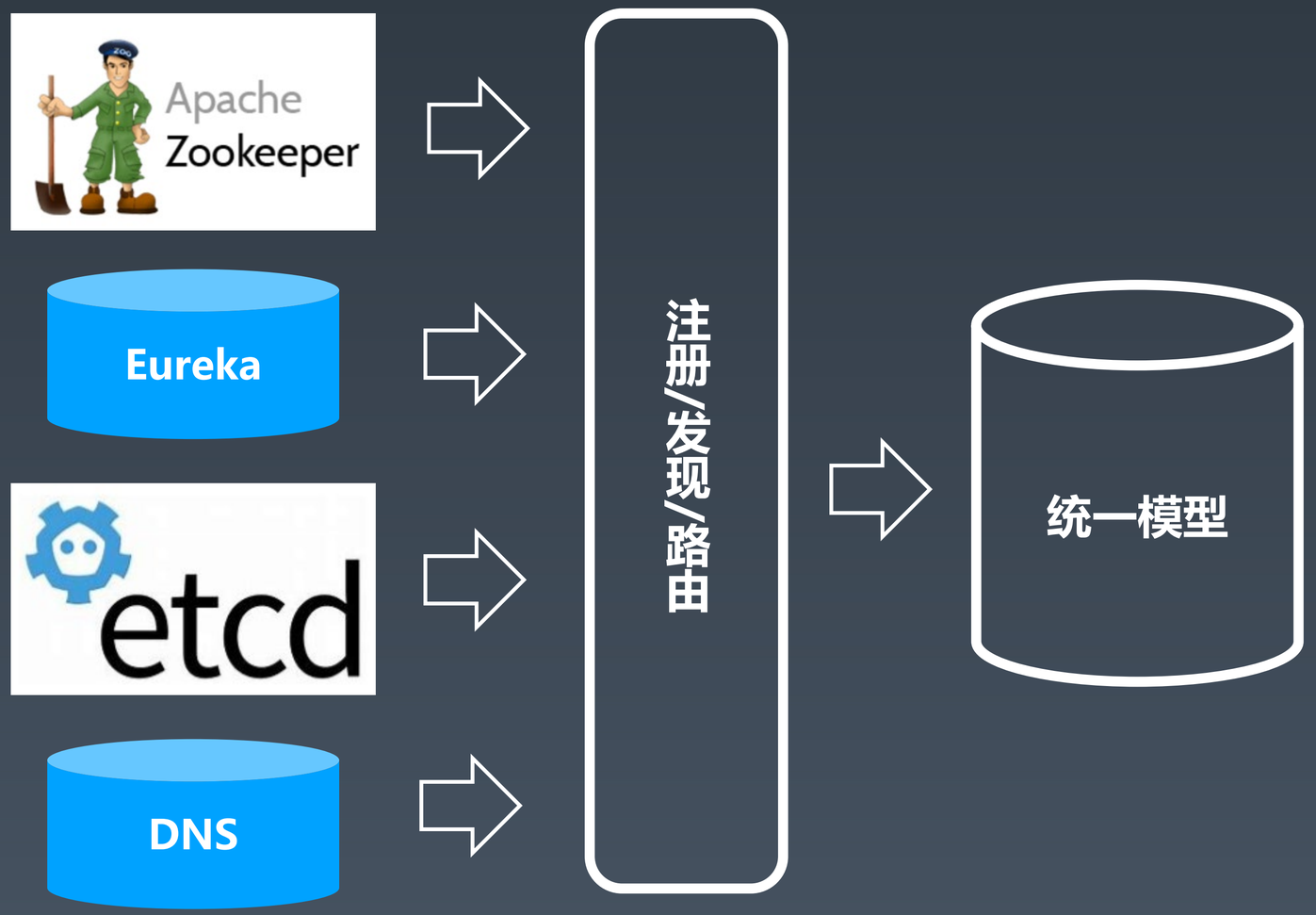
- 兼容既有的知识体系
- 配置不需要变成一门新的“语言”
- No *over* configuration
- 一致的方式

Tips：

- ✓ Ansible具有很好的跨平台能力
- ✓ 脚本化、版本化、留日志



跨微服务平台：Spring，Dubbo，K8s



问题：

- 不同的配置中心，信息一致化的管理
- 私有网络内的互通
- 网络之间的互通
- 跨集群的访问控制
- 协议与数据格式

Tips：

- ✓ 全局规划*内部域名*
- ✓ 用域名，但是不用域名服务器
- ✓ 微服务注册信息->路由规则

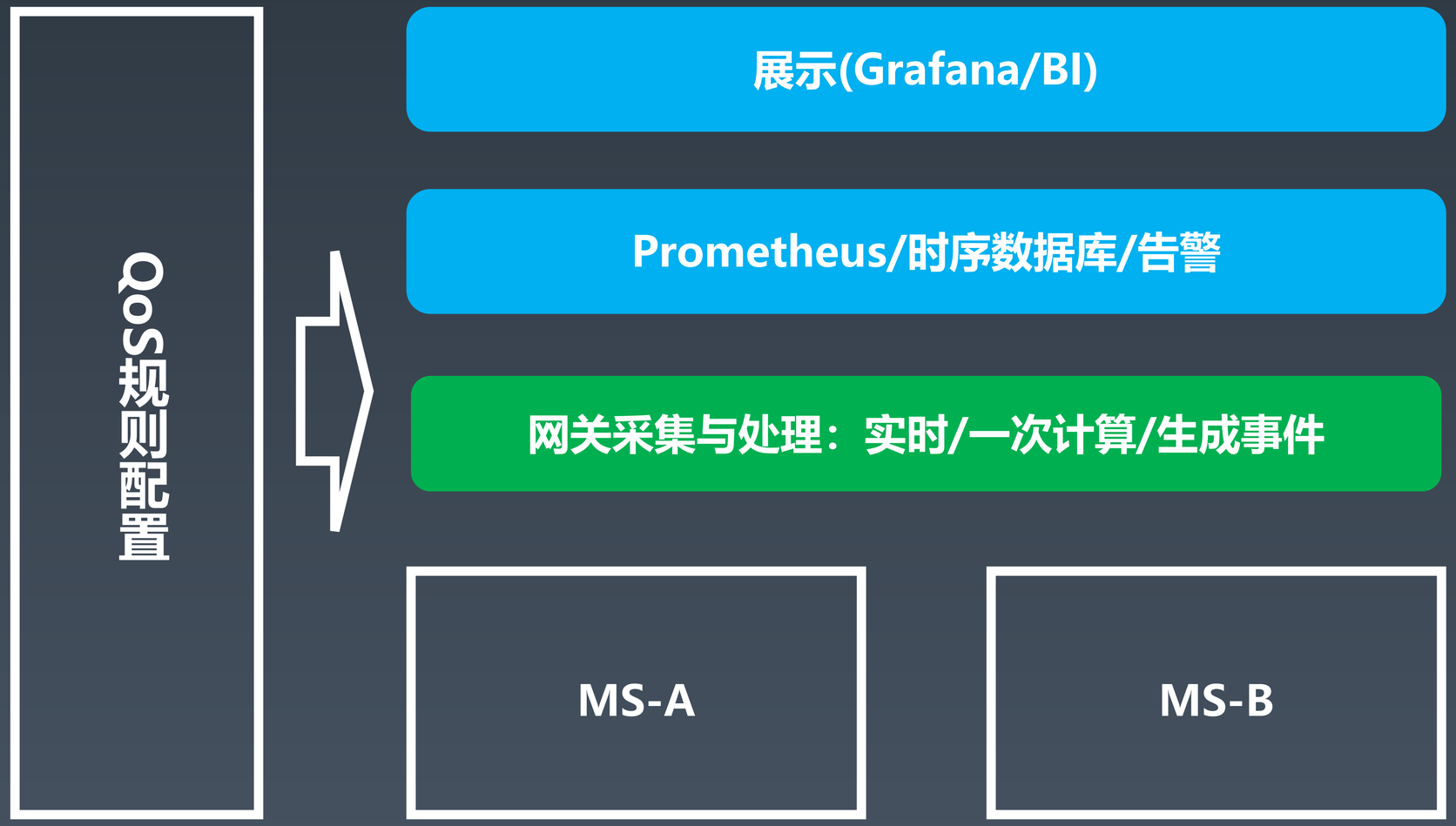
监控与服务质量

核心：

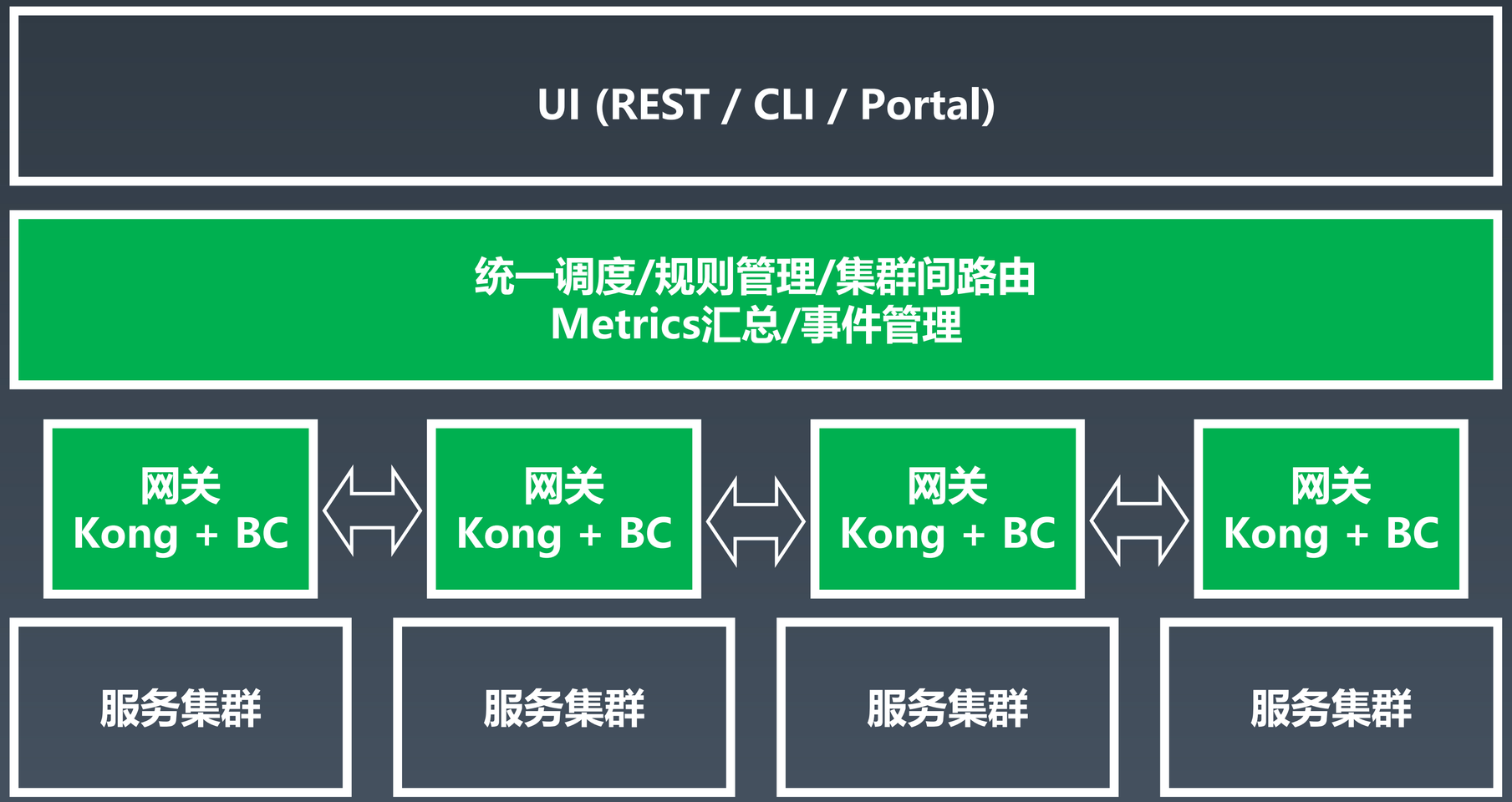
- 服务质量是*目的*
- 度量、监控、治理
- 微管理与宏观管理结合
- 海量指标、实时监控

TIPS：

- ✓ Metrics的推拉结合处理
- ✓ 分层采集、分层处理
- ✓ 微缩的“边缘计算”



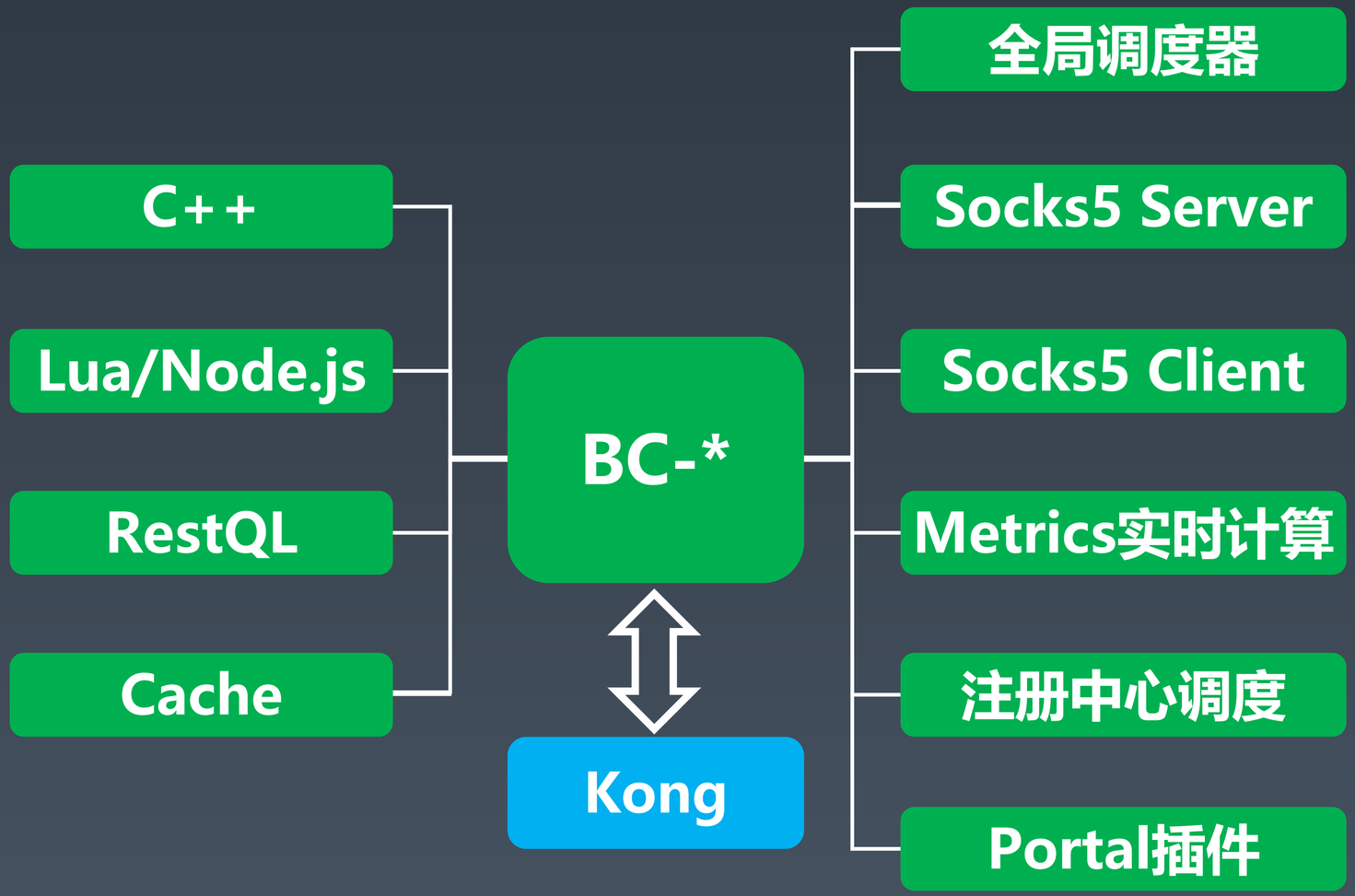
整体拓扑



组件：

- Kong：HTTP流量管理
- BC-PX：非HTTP流量管理
- BC-MON：实时监控
- BC：统一调度
- BC-BOT：自动化配置
- Prometheus：汇总监控
- Grafana / BI：展示
- Liferay / Drupal：门户
- PostgreSQL：All-in-one DataStore

BC-* —— 应用流量管理套件




```
$java -Dhttp.proxyHost
$java -DsocksProxyHost
$bc-px java ...
```

安全——Sec First

应用层安全的采集和实施点

- 全链路加密（如果需要）
- 字段加解密（如果需要）
- 每段链路的身份验证（如果需要）
- 每段链路的访问控制（如果需要）
- 安全事件识别与告警（暴力破解、薅羊毛...）
- 基于审批的工作流

性能

- HTTP Keep-Alive：长连接有效保障了网关间数据传输效率
- Socks5：多一次建立连接操作；连接建立后，性能无损耗
- 流式的格式转换
- 策略、路由和配置信息在网关内存缓存
- 无阻塞操作
- 非控制操作采用旁路处理
- 可以基于域名路由，但是不做域名解析

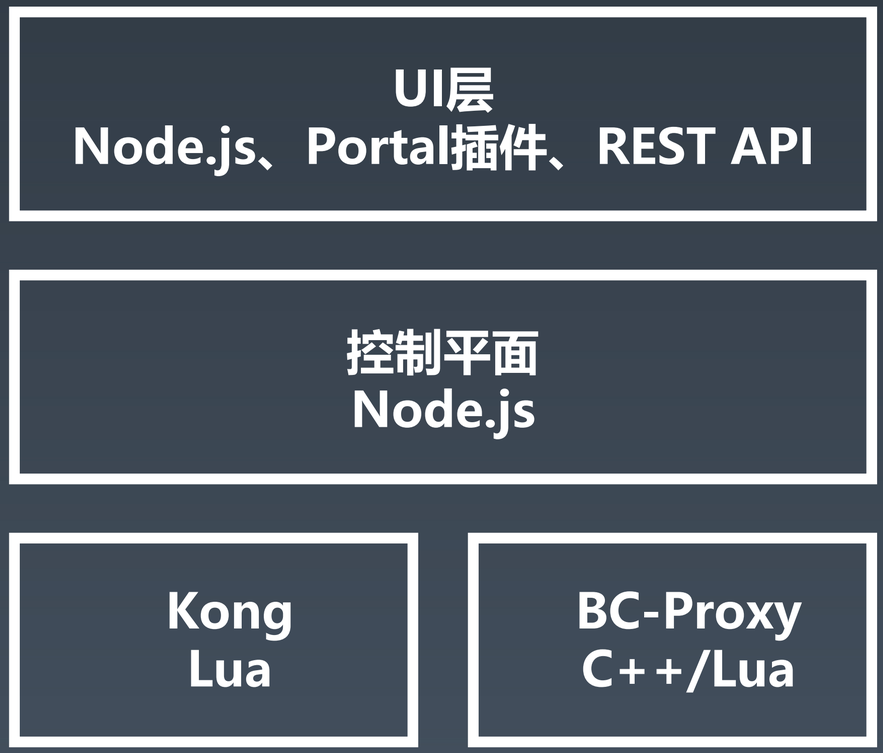
扩展与扩容

扩展：

- C++扩展模块 - 协议适配、编解码、格式
- Lua脚本——code snippet
- Node.js扩展API — Directus
- Node.js呈现 — Gatsby
- 监控集成 - Prometheus
- Tracing集成 - zipkin，jeager, skywalking
- Portal集成 - Liferay, Drupal
- IDM集成

扩容：

Ansible， 加节点，加集群



Geekbang 极客邦科技 | TGO 鲲鹏會

TGO鲲鹏会

汇聚全球科技领导者的高端社群

全球12大城市

850+高端科技领导者

使命 Mission

为社会输送更多优秀的科技领导者

愿景 Vision

构建全球领先的有技术背景优秀人才的学习成长平台

扫描二维码，了解更多内容

THANKS!

QCon 10th