分布式系统的本质

应用整体监控

- **基础层监控**
  OS、主机、网络…
- **中间件层监控**
  消息队列、缓存、数据库、应用容器、网关、RPC框架、JVM…
- **应用层监控**
  API请求、吞吐量、响应时间 、错误码、SQL语句、调用链路、函数调用栈、业务指标…

资源/服务调度

- **计算资源调度**
  CPU，内存、磁盘、网络…
- **服务调度**
  服务编排、服务复本、服务容量伸缩、故障服务迁移、服务生命周期管理…
- **架构调度**
  多租户、架构版本管理、架构部署、运行、更新、销毁管理、多租户管理、灰度发布…

状态/数据调度

- **数据可用性**
  多复本保存
- **数据一致性**
  读写一致性策略
- **数据分布式**
  数据索引、分片

流量调度

- **服务治理**
  服务发现、服务路由、服务降级、服务熔断、服务保护…
- **流量控制**
  负载均衡、流量分配、流量控制、异地灾备 …
- **流量管理**
  协议转换、请求校验、数据缓存、数据计算…

全栈监控技术说明

应用层服务监控

HTTP 请求访问	Java服务 性能监控	JDBC 性能监控	外部服务 调用性能	服务调用 链监控

平台层软件监控

网关 Nginx	Java容器 Tomcat	缓存服务 Redis	消息队列 Kafa	数据库 MySQL

基础机器资源监控

CPU 使用率	内存 使用率	网络 吞吐量	硬盘 I/O吞吐	硬盘 使用率

日志收集

图表事件展示

报表	性能	事件

数据分析

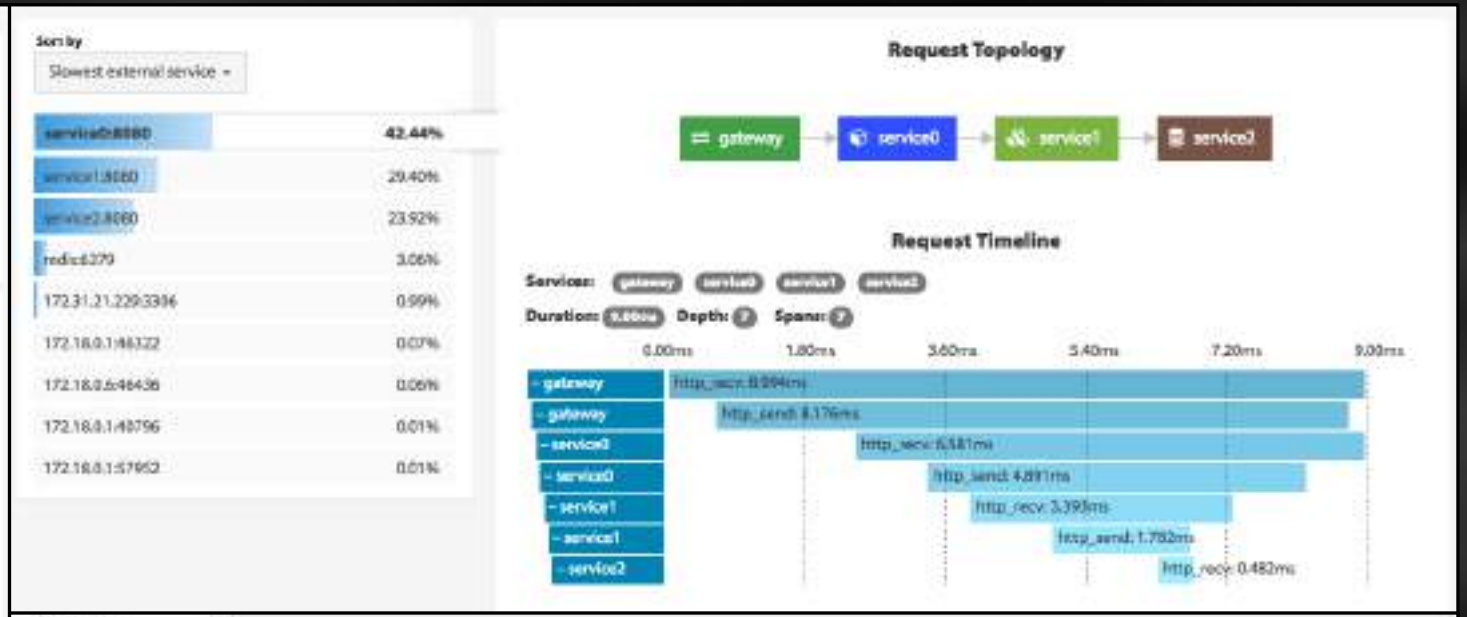
聚合	过滤	计算

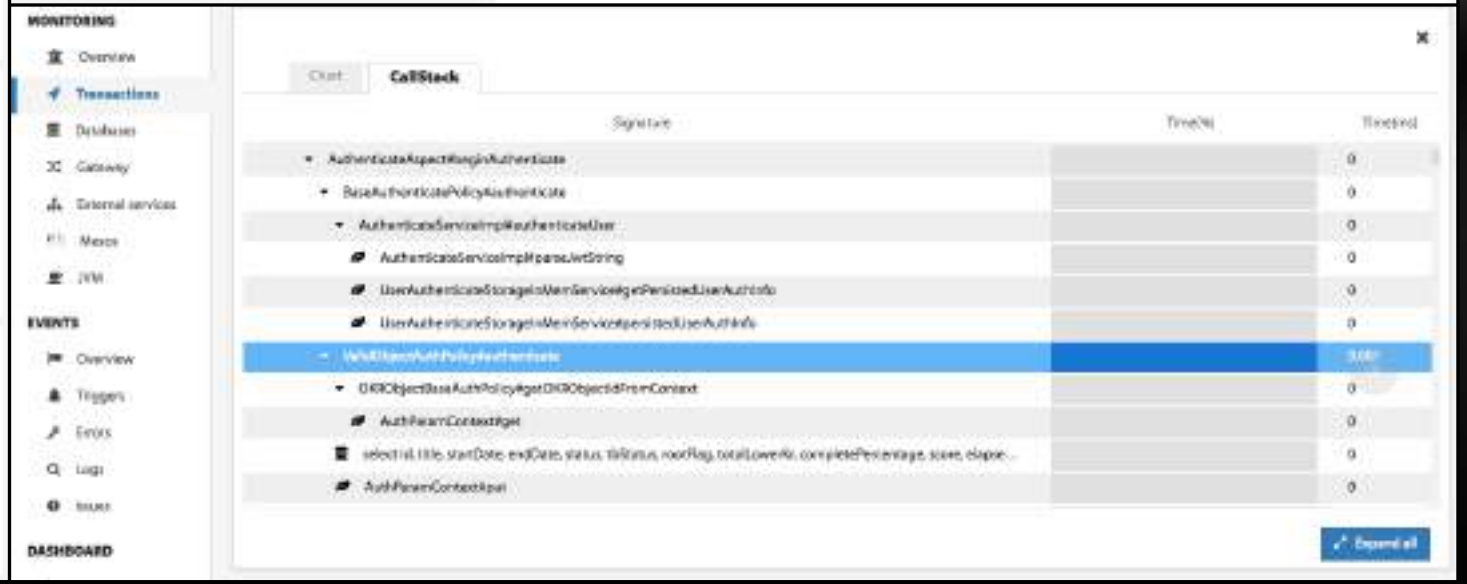
数据存储

指标	日志	规则

MegaEase


System Health Score
System Key Index Overview
31
2,028
173.97
Throughput(RPM)
System Topology
Top Transactions
HTTP Status Code Distribution
Servers
Request Error EMMA (911 Rate)


Sort by
Slowest external service
42.44%
29.40%
23.92%
3.06%
0.99%
0.07%
0.06%
0.01%
0.01%
Request Topology
gateway
service0
service1
service2
Request Timeline
Services:
Durations:
Depth:
Spans:
0.00ms
1.80ms
3.60ms
5.40ms
7.20ms
9.00ms
gateway
gateway
service0
service0
service1
service1
service2


MONITORING
Overview
Transactions
Databases
Gateway
External services
Mbeans
JVM
EVENTS
Overview
Triggers
Errors
Logs
Issues
DASHBOARD
Chart
CallStack
Signature
Time(%)
Expand all

MegaEase

一栈式调度技术说明

Stack 定义

- 描述定义
- 相关组件
- 环境配置
- 集群描述
- 依赖关系
- 健康检查
- 部署方式
- 结点存储

Stack 运行实例

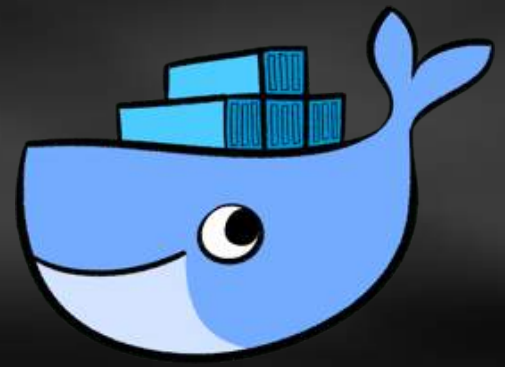
基础物理层

- 服务部署
- 服务发现
- 服务路由（负载均衡）
- 服务健康检查
- 服务调度（伸缩/故障迁移）
- 物理层资源管理
- 服务监控

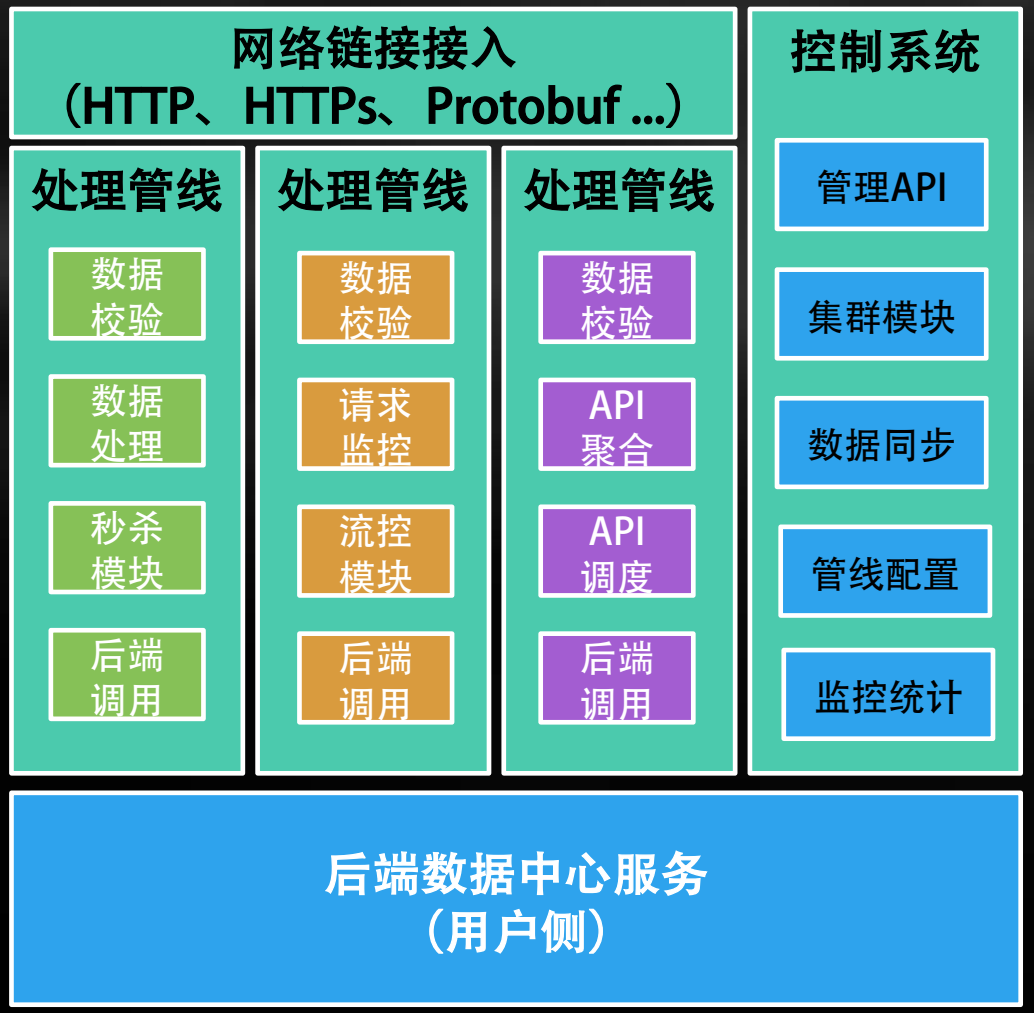
MegaEase

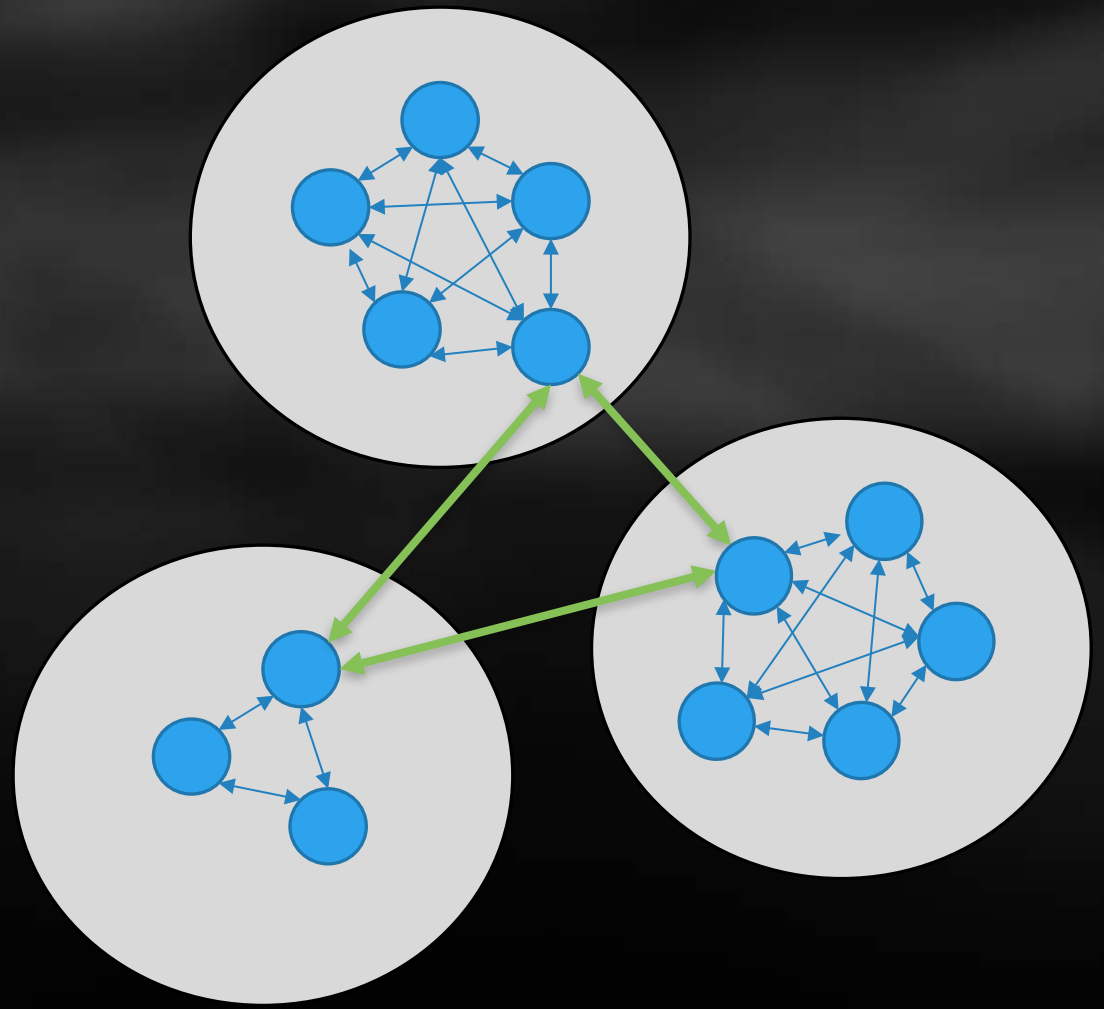
面向服务的Docker镜像

- **为什么要做服务化的Docker镜像**
  - 想想现有的中间件 – nginx 和 zookeeper
  - 都不是服务化的，任何静态配置或动态配置的改变都需要通过改conf文件
- **面对不完美世界的Workaround的方案**
  - 我们开发了一个服务化的Docker EntryPoint框架，提供如下最基本回调脚本
  - Start， Stop， ApplyConfig，HealthChecking



网关流量调度技术





如何设计一个好的流量调度网关

- **现在的所有的Gateway在设计理念上都达不我们想要的高度。**
  - Nginx和OpenResty明显就是给运维人员用的。
- **一个好的Gateway需要有如下的特性**

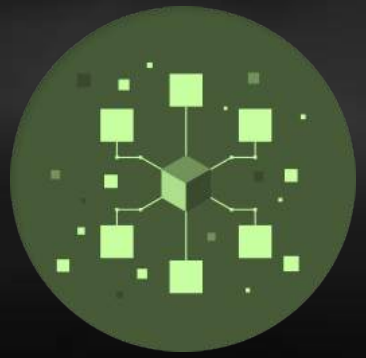
Service	Cluster	Lambda
• 首先一定要是一个Service，作为service的一个标志是：通过API去改配置，而不是通过文件。	• 其次，还要是一个Cluster，作为Cluster的一个标志是：能够互相复制数据，可以集群内分组，我们使用NRW模型+Gossip/Raft协议。	• 最后，还可入嵌入用户代码，既 Serverless 或 AWS Lambda。

状态/数据调度

Google App Engine的co-founder Ryan Barrett
2009年的google i/o上的演讲《Transaction Across DataCenter》
（视频： http://www.youtube.com/watch?v=srOgpXECblk）

MegaEase

状态/数据调度 - 分布式存储系统



Data Scheme

关系型数据库、NoSQL
时序数据库、搜索数据库
OLAP/OLTP、漏洞数据……

数据格式太多，很难做出放之四海皆准的

Data Storage

廉价的：开源方案NFS/Ceph/TiDB/Cockroach…
中间价：云产商方案 S3/Dynamo/Aurora…
昂贵的：EMC、 Nutanix…

不同的分布式存储方案有不同优缺点

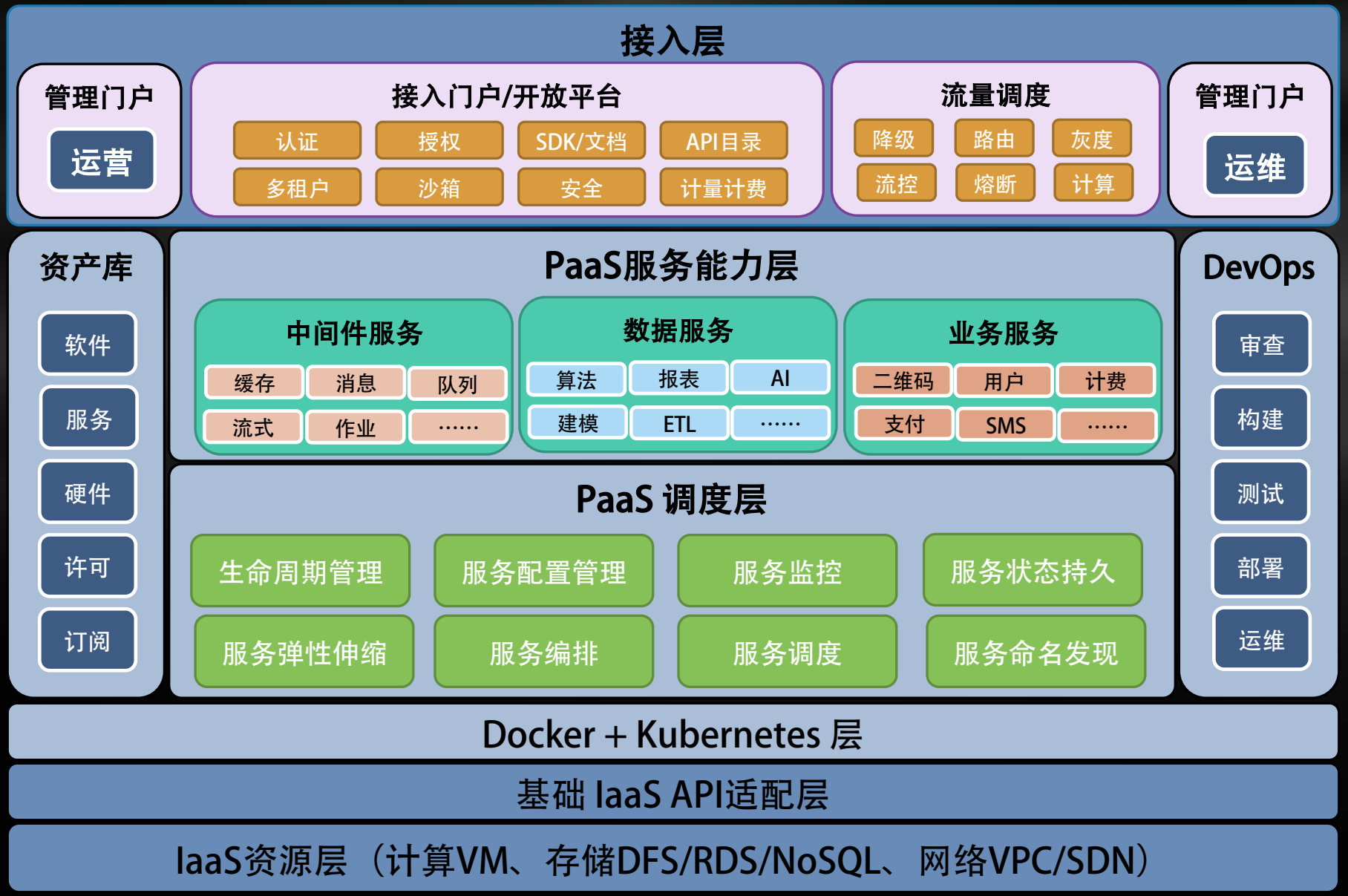
PaaS平台是什么样的

MegaEase

PaaS平台的重要性

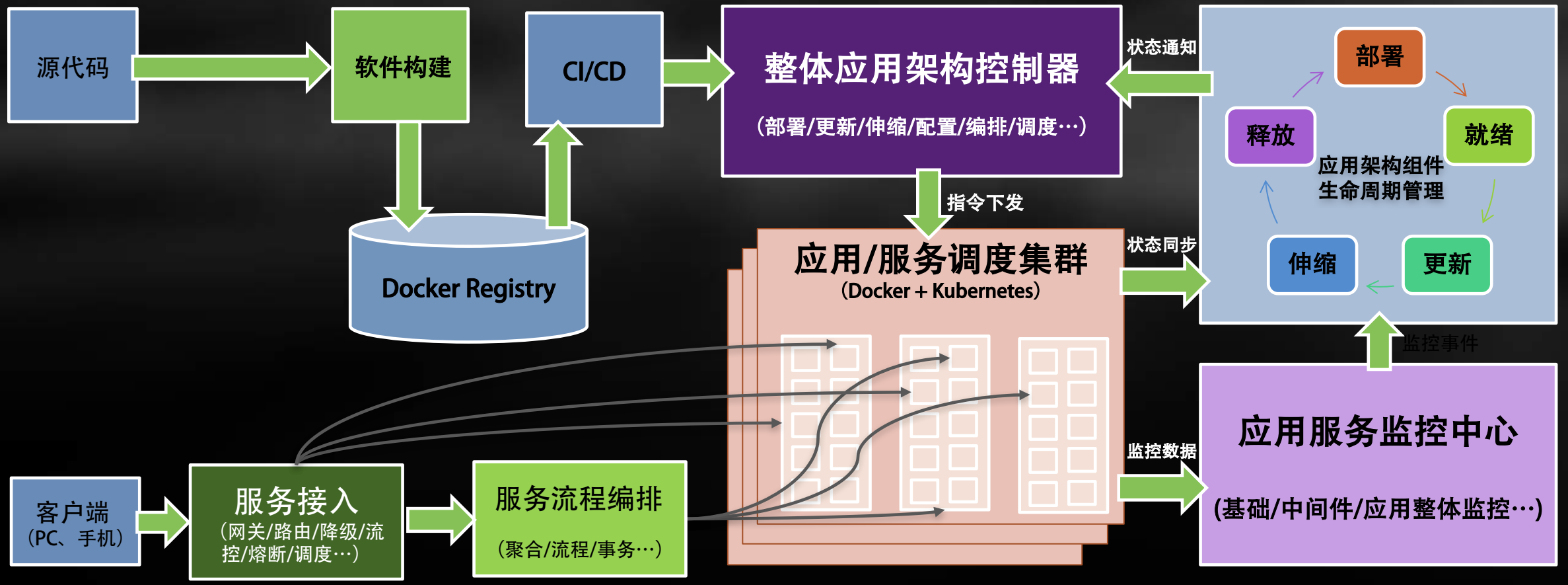
- 软件工程的三个核心
  - 服务SLA
    - 高可用的系统
    - 自动化的运维
  - 能力和资源重用
    - 软件模块的重用
    - 软件运行环境和资源的重用
  - 过程自动化
    - 软件生产流水线
    - 软件运维自动化

- 这三个核心能力全部体现在PaaS上
  - 分布式多层的系统架构
  - 服务化的能力供应
  - 自动化的运维能力


接入层
管理门户
运营
接入门户/开放平台
认证
授权
SDK/文档
API目录
多租户
沙箱
安全
计量计费
流量调度
降级
路由
灰度
流控
熔断
计算
管理门户
运维
资产库
软件
服务
硬件
许可
订阅
PaaS服务能力层
中间件服务
缓存
消息
队列
流式
作业
……
数据服务
算法
报表
AI
建模
ETL
……
业务服务
二维码
用户
计费
支付
SMS
……
PaaS 调度层
生命周期管理
服务配置管理
服务监控
服务状态持久
服务弹性伸缩
服务编排
服务调度
服务命名发现
DevOps
审查
构建
测试
部署
运维
Docker + Kubernetes 层
基础 IaaS API适配层
IaaS资源层（计算VM、存储DFS/RDS/NoSQL、网络VPC/SDN）

MegaEase

开发/运维示意图



MegaEase

创业的相关产品介绍

MegaEase 的三个产品

Ease Monitor

基于ELK完全开放式的APM

- **基础层监控**
  OS、主机、网络…
- **中间件层监控**
  消息队列、缓存、数据库、应用容器、网关、RPC框架、JVM…
- **应用层监控**
  API请求、吞吐量、响应时间 、错误码、SQL语句、调用链路、函数调用栈、业务指标…

Ease Stack

基于 Kubernetes 的二级调度

- 一键全栈部署
  一次架构定义、多次灵活部署
- 容量弹性伸缩
  手动或自动的管理应用集群容量
- 动态配置更新
  集中管理、自动下发
- 应用状态保持
  持久应用状态、自动管理关联关系

Ease Gateway

自主研发的服务化集群式的api网关

- **集群化**
  Gossip/RAFT协议，NRW模型、集群分组
- **服务化**
  管理API、统计数据……
- **插件式**
  - 用户逻辑插入，Lambda

MegaEase


谢谢