1024040905吴传智

目 录 CONTENTS

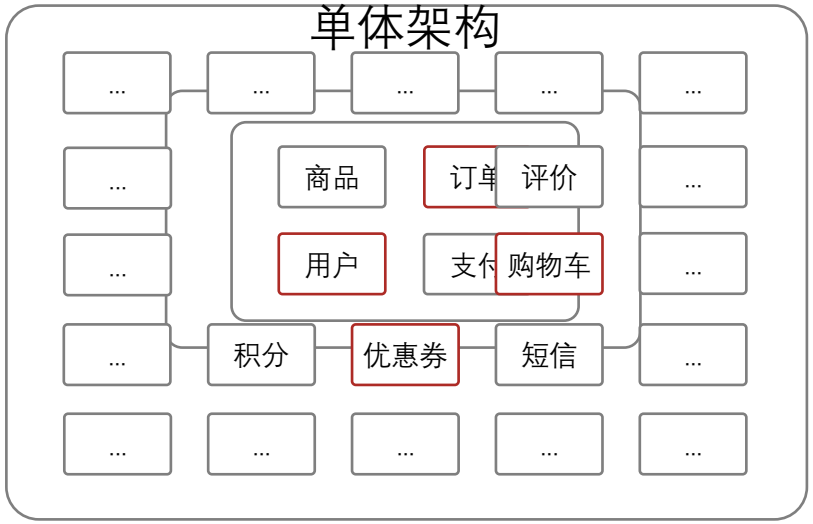
01

微服务技术

微服务

微服务是一种软件架构风格，它是以专注于单一职责的很多小型项目为基础，组合出复杂的大型应用。

单体架构



维基百科
自由的百科全书

微服务是一种软件架构风格，它是以专注于单一职责的很多小型项目为基础，组合出复杂的大型应用。

单体架构

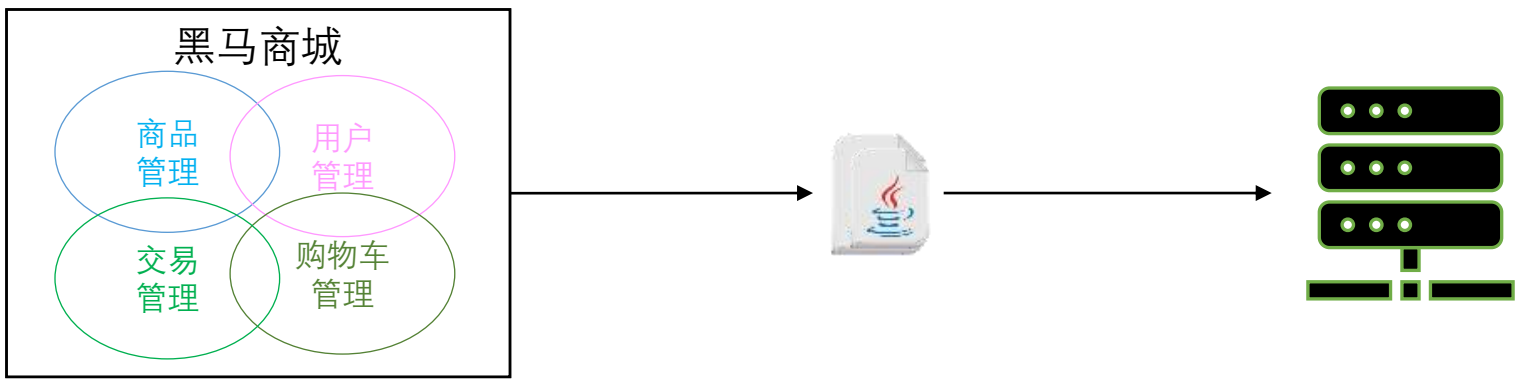
单体架构：将业务的所有功能集中在一个项目中开发，打成一个包部署。

优点：

- 架构简单
- 部署成本低

缺点：

- 团队协作成本高
- 系统发布效率低
- 系统可用性差

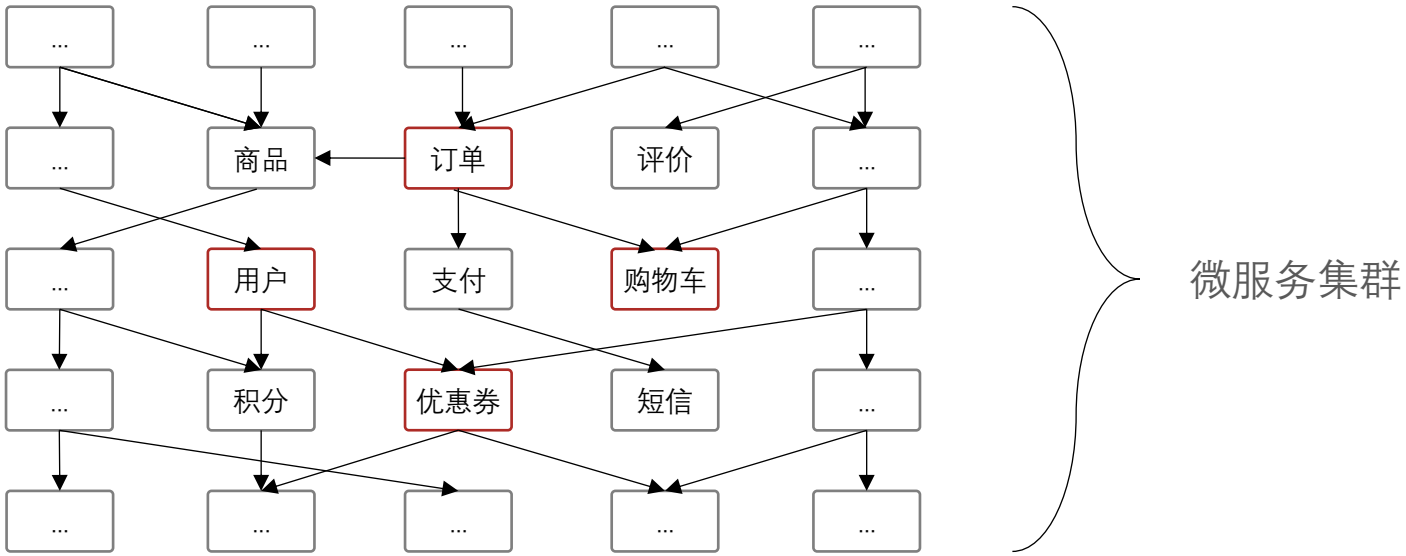


总结：

单体架构适合开发功能相对简单，规模较小的项目。

微服务

- 服务拆分
- 远程调用
- 服务治理
- 请求路由
- 身份认证
- 配置管理
- 服务保护
- 分布式事务
- 异步通信
- 消息可靠性
- 延迟消息
- 分布式搜索
- 倒排索引
- 数据聚合

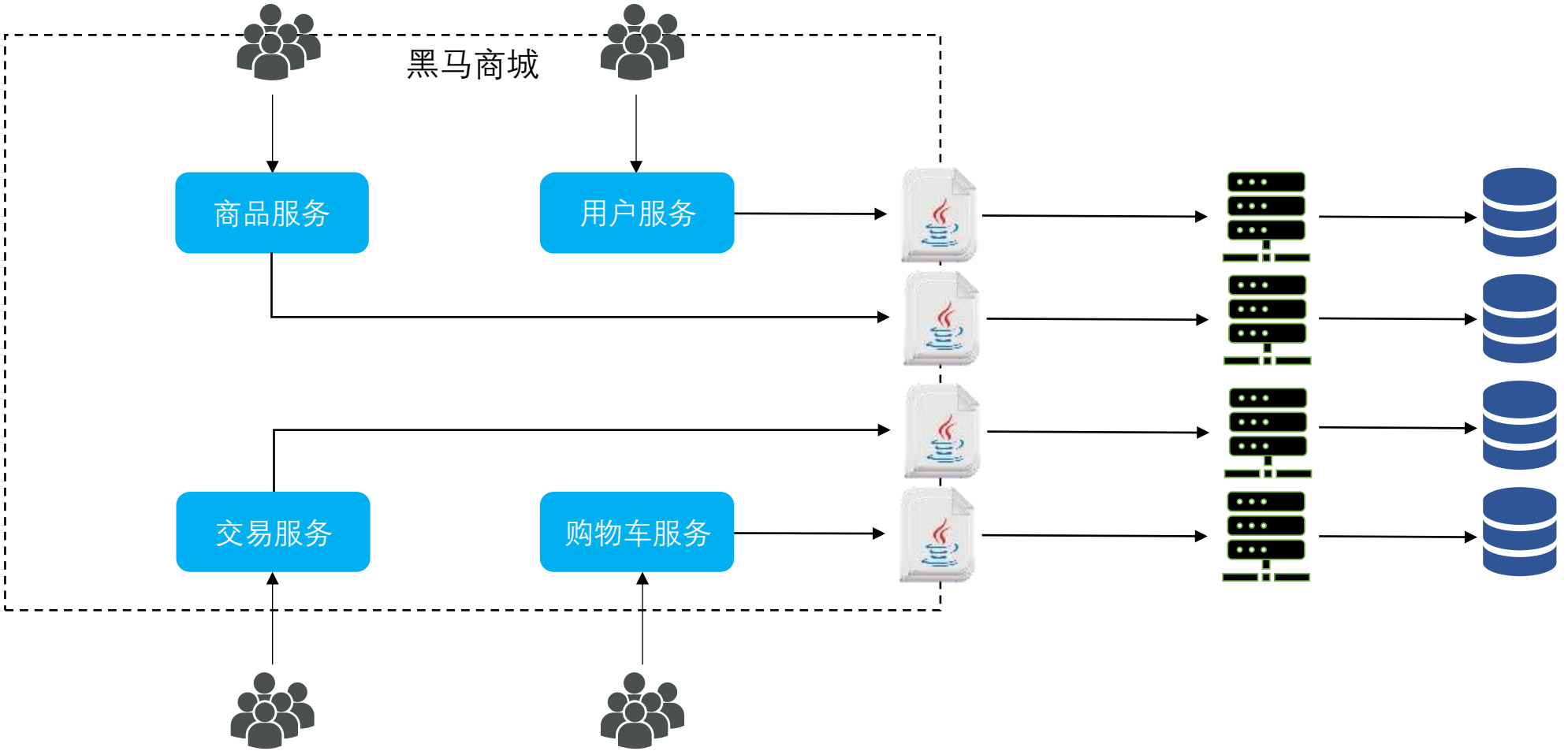


微服务是一种软件架构风格，它是以专注于单一职责的很多小型项目为基础，组合出复杂的大型应用。

微服务

微服务架构，是服务化思想指导下的一套最佳实践架构方案。服务化，就是把单体架构中的功能模块拆分为多个独立项目。

- 粒度小
- 团队自治
- 服务自治



SpringCloud

SpringCloud是目前国内使用最广泛的微服务框架。官网地址：https://spring.io/projects/spring-cloud。

SpringCloud集成了各种微服务功能组件，并基于SpringBoot实现了这些组件的自动装配，从而提供了良好的开箱即用体验：

服务注册发现

Eureka、Nacos、Consul

服务远程调用

OpenFeign、Dubbo

服务链路监控

Zipkin、Sleuth

统一配置管理

SpringCloudConfig、Nacos

统一网关路由

SpringCloudGateway、Zuul

流控、降级、保护

Hystix、Sentinel

02

DevOps

定义及起源与发展

01

DevOps的定义

DevOps是Development和Operations的组合，是一种重视软件开发人员（Dev）和运维人员（Ops）之间沟通、协作与整合的文化、方法和实践。它打破了传统的“开发”与“运维”之间的壁垒，通过自动化、持续集成、持续交付、监控与反馈等手段，实现软件从开发到部署、从运行到维护的全流程高效协同

02

DevOps的起源与发展

DevOps起源于敏捷软件开发方法论，旨在解决开发和运维之间的隔阂问题。其发展历程经历了萌芽阶段、快速成长阶段和成熟阶段，目前已广泛应用于各个领域，并随着云计算、容器化技术和微服务架构的发展不断演进

DevOps的核心理念

•持续集成与持续交付：开发者频繁地将代码集成到主干中，并通过自动化测试来确保代码质量。持续交付则是在持续集成的基础上，进一步实现软件的自动化部署 。

• 基础设施即代码：使用代码化的方式来管理和配置基础设施资源，使得基础设施的管理变得更加灵活和自动化，减少了人为错误，提高了环境的一致性 。

• 监控与日志记录：通过实时监控系统性能和收集日志数据，及时发现和解决潜在问题，确保系统的稳定运行 。

• 自动化测试：编写自动化测试用例，在每次代码变更时进行全面的测试，从而快速发现和修复缺陷 。

• 文化和协作：强调团队之间的协作和透明度，鼓励开发者和运维人员共同承担责任，推动持续改进 。

DevOps的技术链

- 代码管理（SCM）：GitHub、GitLab、BitBucket、SubVersion
- 构建工具：Ant、Gradle、maven
- 自动部署：Capistrano、CodeDeploy
- 持续集成（CI）：Bamboo、Hudson、Jenkins
- 配置管理：Ansible、Chef、Puppet、SaltStack、ScriptRock GuardRail
- 容器：Docker、LXC、第三方厂商如AWS
- 编排：Kubernetes、Core、Apache Mesos、DC/OS
- 服务注册与发现：Zookeeper、etcd、Consul
- 脚本语言：python、ruby、shell
- 日志管理：ELK、Logentries
- 系统监控：Datadog、Graphite、Icinga、Nagios
- 性能监控：AppDynamics、New Relic、Splunk
- 压力测试：JMeter、Blaze Meter
- 预警：PagerDuty、pingdom、厂商自带如AWS SNS
- HTTP加速器：Varnish
- 消息总线：ActiveMQ、SQS
- 应用服务器：Tomcat、JBoss
- Web服务器：Apache、Nginx、IIS
- 数据库：MySQL、Oracle、PostgreSQL等关系型数据库；cassandra、mongoDB、redis等NoSQL数据库
- 项目管理（PM）：Jira、Asana、Taiga、Trello、Basecamp、Pivotal Tracker

03

移动云容器服务KCS

容器服务KCS原理介绍

移动云容器服务 KCS 的原理基于 Kubernetes 技术体系，通过 Master 节点和 Node 节点协同工作，利用核心概念如 Pod、Service、Label 等实现应用的部署、调度、运行和管理。在网络通信方面采用 CNI 插件和 DNS 服务发现机制，支持多种存储卷类型并可自动或手动进行弹性伸缩，为用户提供高效、灵活、可扩展的容器化应用管理平台。

容器服务KCS功能介绍

容器服务基于Kubernetes技术，依托移动云基础资源，构建高性能、可扩展的Kubernetes集群，为用户容器应用提供一站式生命周期管理，助力用户应用轻松上云。通过云原生AI套件，为客户提供强大的AI任务管理及加速能力。

容器服务KCS能力优势

简单易用

一键创建容器集群，一站式部署、运维容器化应用，实现应用全生命周期管理

安全可靠

支持VPC隔离；配套容器安全产品，支持容器镜像安全、容器运行时安全；

快速创建

百节点托管集群创建仅需10min

集群规模

单个集群可管理节点数量达千级

智算原生

支持Nvidia、华为、天数、壁仞等多种异构GPU混合管理，并支持万卡级资源调度

开放兼容

Kubernetes服务提供商资质（KCSP）
完全兼容Kubernetes（CNCF一致性认证）
可信云全栈容器云解决方案

容器服务KCS应用场景

口 某科技公司推理业务

该公司运营着多个业务系统，涵盖语音识别、图像分析、智能推荐等领域，这些系统不仅需要处理大量的实时数据，还要求快速响应和高精度的推理能力。

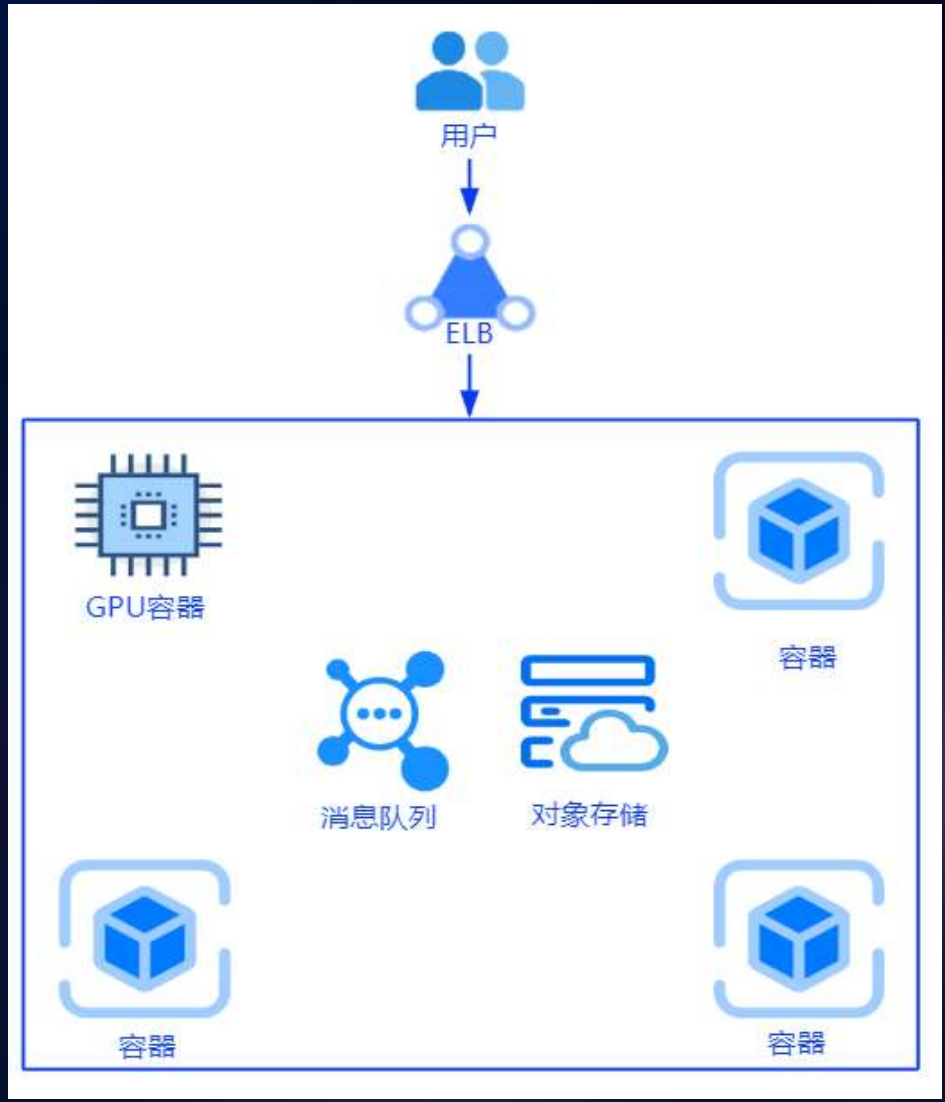
KCS对场景的支持解析：

① 所有业务系统均以容器化部署在KCS中，确保系统在面对流量峰值时能迅速调整资源，维持稳定的服务质量。

② 针对图像分析等高计算需求的推理业务，部署GPU容器加速了深度学习模型，推理任务的处理速度获得了显著提升。

③ 支持不同业务模块之间的异步消息传递，如Kafka或RocketMQ。

推荐搭配：容器镜像+ 容器服务+云负载均衡+消息队列+对象存储



容器服务KCS应用场景

□ 微服务应用管理

微服务架构应用能方便的实现产品敏捷开发、快速迭代的需求。

KCS对场景的支持解析：

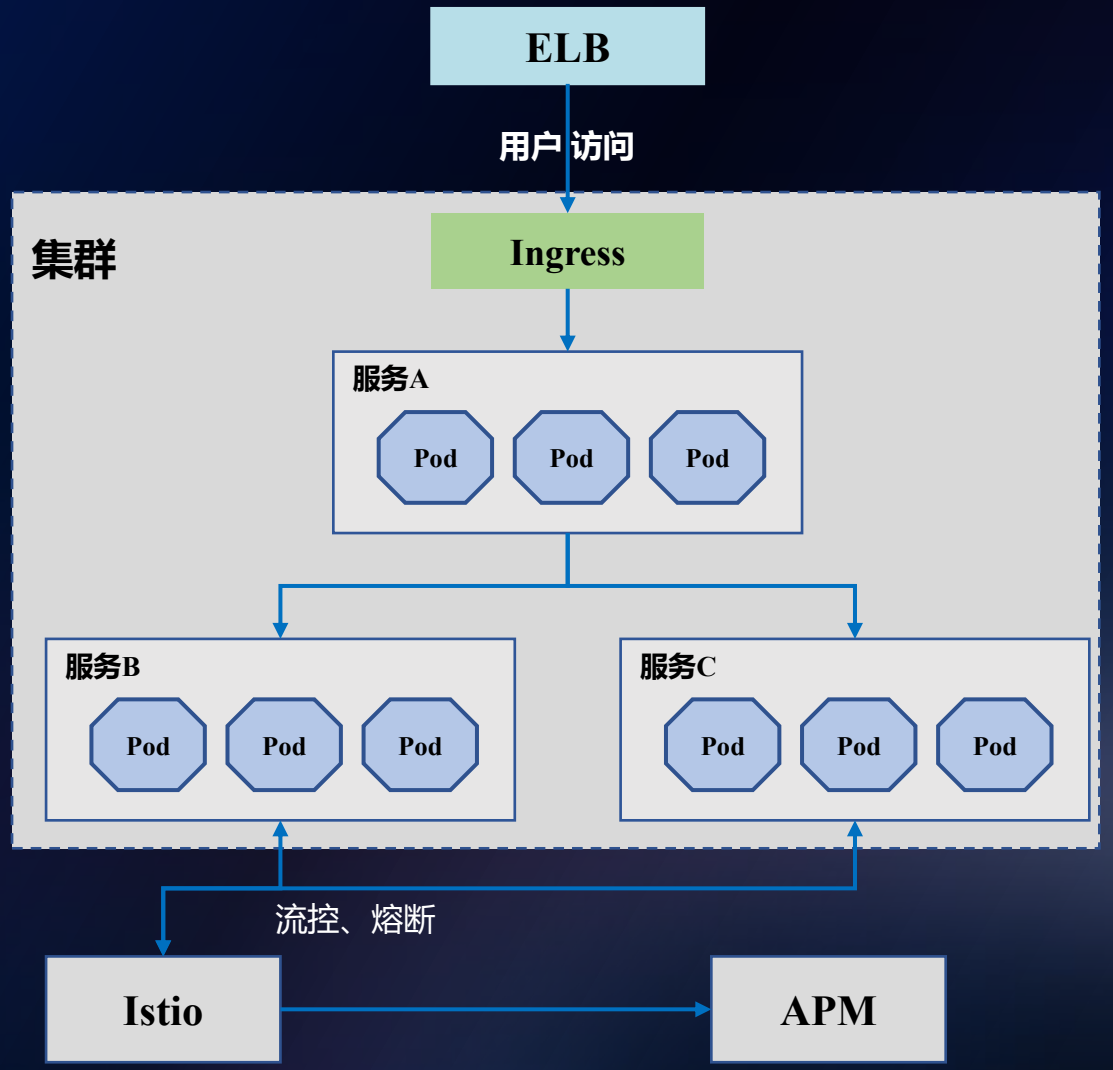
① 将合理拆分的微服务模块镜像存储在移动云容器镜像仓库，进行全生命周期管理；

② 移动云容器服务产品提供各个微服务的部署和全生命周期管理；

③ 搭配微服务治理产品，支持微服务治理，可提供应用进行流控、灰度、熔断等治理能力

④ 支持应用的灰度发布；

⑤ 支持Helm的应用部署



推荐搭配：容器镜像+ 容器安全+容器服务+云负载均衡

容器服务KCS应用场景

口 电商/直播业务

针对电商业务、视频直播业务等具有明显业务峰谷值波动的行业，传统应用部署和运行方式不能灵活的进行业务扩缩容的动态调整适应流量变化。容器服务则可以根据业务流量进行自动扩容/缩容，无需人工干预，能够有效避免流量激增扩容不及时导致的系统崩溃，以及平时大量资源闲置造成的浪费。

KCS对场景的支持解析：

① 容器业务Pod实例的手动扩缩容

② 根据资源利用率（CPU/内存）、时间定义自动弹性伸缩策略，秒级触发容器的伸缩操作。

③ 集群的自动化弹性扩容

推荐搭配：云主机+云硬盘+负载均衡器+容器镜像+容器服务

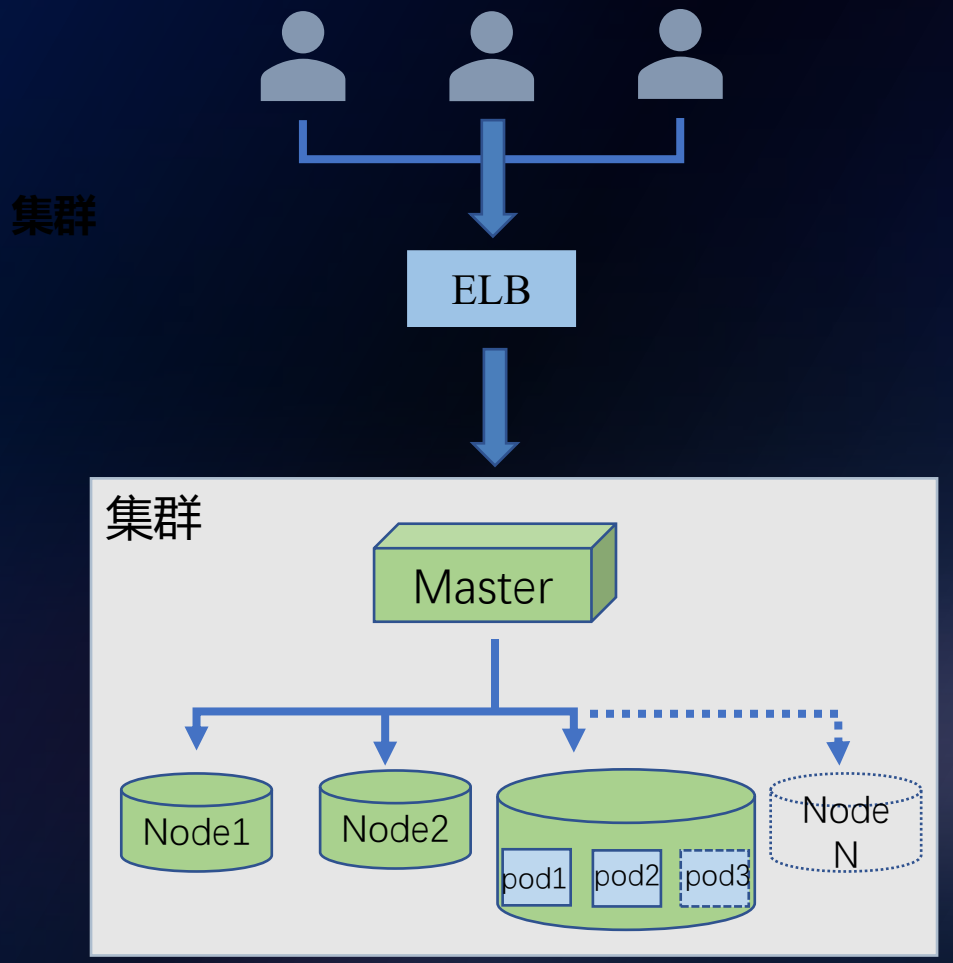
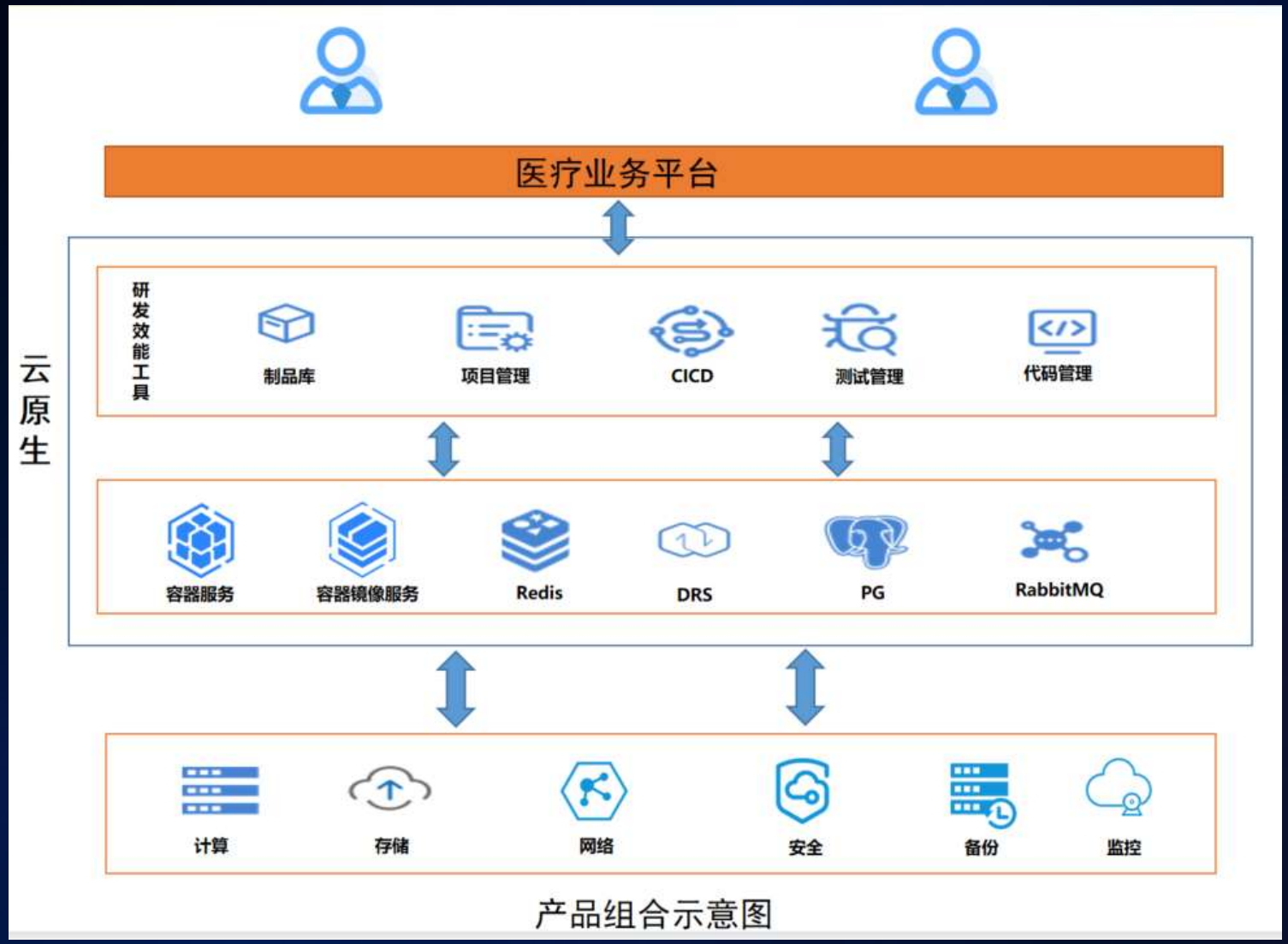


图2-弹性扩容

容器服务KCS案例实践

移动云

某医疗行业客户



产品组合示意图

某医疗企业在国内主要业务为采集人类遗传资源组建基因库，该企业处于快速发展阶段，产品迭代频率需要小步快跑，敏捷发版优化体验，业务量快速大幅增长，资源扩容需求较为紧迫。针对客户需求，为客户打造了以移动云容器服务、数据库为业务承载核心产品，搭配容器镜像、消息队列以及研发效能工具构建DevOps敏捷交付链，并配合安全、监控等产品的一整套解决方案。

容器服务提供专有集群用于业务部署，3台master节点保障管理面高可用；业务节点跨多可用区，集群核心组件监控和健康巡检7X24小时监测集群健康状态；集群、节点、应用、容器多层级资源监控随时掌握业务负载情况；弹性伸缩实现按需调度资源，全面提升资源使用率；

容器服务KCS案例实践

移动云

- 某旅游行业客户

某旅游局自2020年12月订购使用容器服务集群，已持续平稳运行18个月。

承载了景区运营分析、预约系统以及景区直播等多个核心业务。其中景区运营分析、预约系统均包括大量重要数据与客户信息，对于平台数据存储和容灾能力都提出了较高要求，直播业务更是对产品时延、负载能力和稳定性发起挑战。

容器服务搭配容器镜像、云主机、文件存储等基础产品构建基础业务承载平台，使用容器安全等产品提供安全保障，形成一套完全匹配客户业务需求的标准解决方案。容器化运行业务，实现了应用发布、回滚的自动化，大幅减少运维工作；业务Pod的弹性伸缩将业务扩展的时间从小时级缩短为秒级。