

NEW PLAN

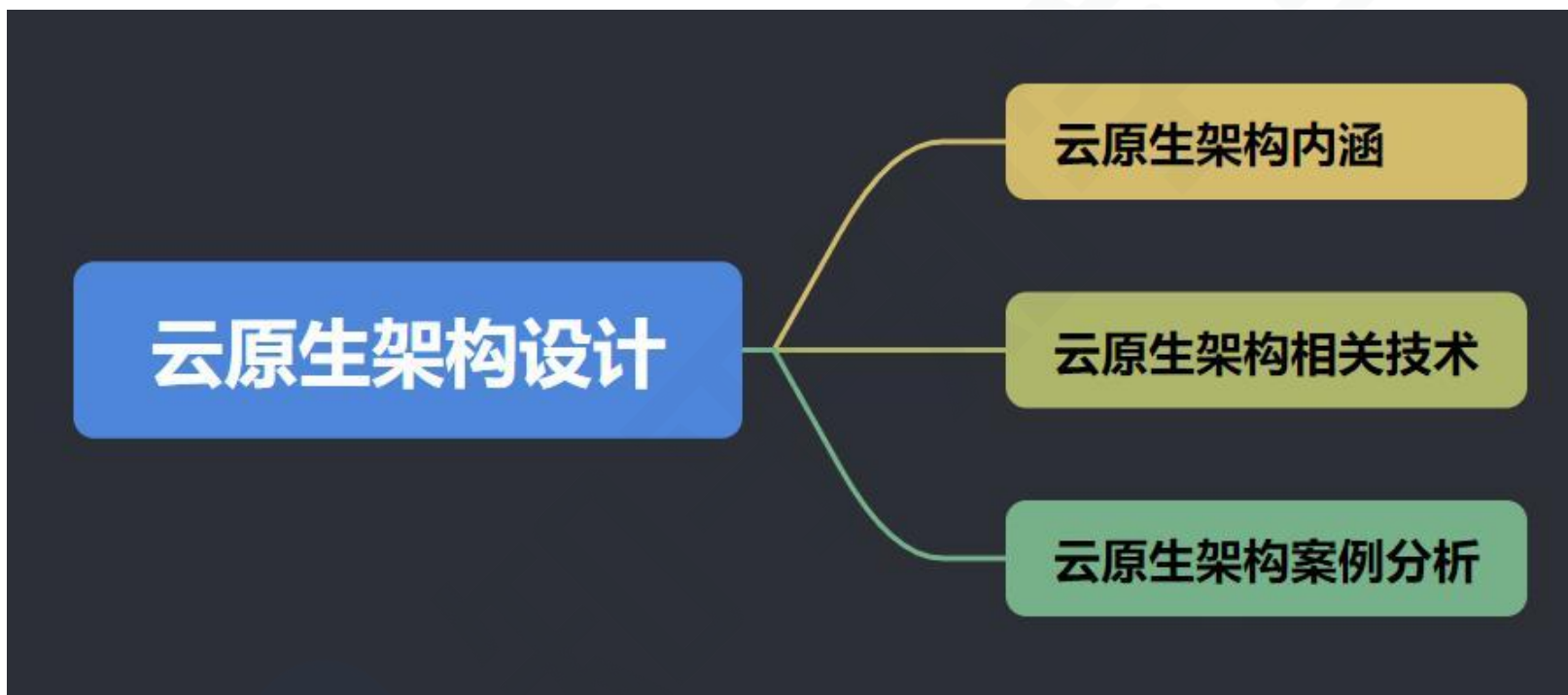
软考高级架构师

• 一段新征程 •





云原生架构设计





云原生架构内涵

- 云原生架构是**基于云原生技术**的一组架构原则和设计模式的集合，旨在将**云应用中的非业务代码部分进行最化的剥离**，从而让云设施接管应用中原有的大量非功能特性（如弹性、韧性、安全、可观测性、灰度等），使业务不再有非功能性业务中断困扰的同时，具备轻量、敏捷、高度自动化的特点。
- 云原生的代码通常包括三部分：业务代码、三方软件、处理非功能特性的代码。从业务代码中剥离大量非功能性特性（不会是所有，比如易用性还不能剥离）到IaaS（基础设施即服务）和PaaS（平台即服务）中。
- 具备云原生架构的应用可以**最大程度利用云服务和提升软件交付能力，进一步加快软件开发**。其特点包括：代码结构发生巨大变化、非功能性特性大量委托、高度自动化的软件交付。
- 云原生架构原则
 - 服务化原则：拆分为微服务架构、小服务架构，分别迭代。
 - 弹性原则：系统的部署规模可以随着业务量的变化而自动伸缩。
 - 可观测原则：通过日志、链路跟踪和度量等手段。
 - 韧性原则：当软件所依赖的软硬件组件出现各种异常时，软件表现出来的抵御能力。
 - 所有过程自动化原则：一方面标准化企业内部的软件交付过程，另一方面在标准化的基础上进行自动化，通过配置数据自描述和面向终态的交付过程。
 - 零信任原则：默认情况下不应该信任网络内部和外部的任何人/设备/系统，需要基于认证和授权重构访问控制的信任基础，以身份为中心。
 - 架构持续演进原则：云原生架构本身也必须是一个具备持续演进能力的架构。



云原生架构内涵

主要架构模式

- **服务化架构模式**：典型模式是**微服务和小服务**模式。通过服务化架构，把**代码模块关系和部署关系进行分离**，每个接口可以部署不同数量的实例，单独扩缩容，从而使得整体的部署更经济。
- **Mesh化架构模式**：把中间件框架（如RPC、缓存、异步消息等）从业务进程中分离，让中间件SDK与业务代码进一步解耦，从而使得中间件升级对业务进程没有影响。分离后在业务进程中只保留很“薄”的Client部分。
- **Serverless模式**：将“部署”这个动作从运维中“收走”，使开发者不用关心应用运行地点、操作系统、网络配置、CPU性能等，也就是把应用的整个运行都委托给云。
- **存储计算分离模式**：在云环境中，推荐把各类暂态数据（如session）、结构化和非结构化持久数据都采用云服务来保存，从而实现存储计算分离。
- **分布式事务模式**：大颗粒度的业务需要访问多个微服务，必然带来分布式事务问题，否则数据就会出现不一致。架构师需要根据不同的场景选择合适的分布式事务模式。
- **可观测架构**：可观测架构包括Logging、Tracing、Metrics 三个方面，其中Logging提供多个级别的详细信息跟踪，由应用开发者主动提供；Tracing提供一个请求从前端到后端的完整调用链路跟踪对于分布式场景尤其有用；Metrics则提供对系统量化的多维度度量。
- **事件驱动架构**：本质上是一种应用/组件间的集成架构模式。可用于服务解耦、增强服务韧性、数据变化通知等场景中



云原生架相关技术

- 容器技术：容器作为标准化软件单元，它将**应用及其所有依赖项打包，使应用不再受环境限制，在不同计算环境间快速、可靠地运行**。通过容器技术，企业可以充分发挥**云计算弹性优势**，降低运维成本。
- Kubernetes已经成为容器编排的事实标准，被广泛用于自动部署，扩展和管理容器化应用。Kubernetes提供了分布式应用管理的核心能力，包括：资源调度、应用部署与管理、自动修复、服务发现与负载均衡、弹性伸缩、声明式API、可扩展性架构、可移植性。
- 云原生微服务：微服务模式将**后端单体应用拆分为松耦合的多个子应用**，每个子应用负责一组子功能。这些子应用称为“微服务”，多个“微服务”共同形成了一个物理独立但逻辑完整的分布式微服务体系。这些微服务相对独立，通过解耦研发、测试与部署流程，提高整体迭代效率。
- 微服务设计约束：
 - 微服务**个体约束**：功能在业务域划分上应是相互独立的，低耦合、单一职责。
 - 微服务与微服务之间的**横向关系**：。主要从微服务的可发现性和可交互性处理服务间的横向关系，一般需要服务注册中心。
 - 微服务与数据层之间的**纵向约束**：在微服务领域，提供数据存储隔离原则，即数据是微服务的私有资产，对于该数据的访问都必须通过当前微服务提供的API来访问。
 - 全局视角下的**微服务分布式约束**：故障发现时效性和根因精确性始终是开发运维人员的核心诉求。



云原生架构相关技术

主要微服务技术：

- Apache Dubbo作为源自阿里巴巴的一款开源高性能RPC框架，特性包括基于透明接口的RPC、智能负载均衡、自动服务注册和发现、可扩展性高、运行时流量路由与可视化的服务治理。
- Spring Cloud作为开发者的主要微服务选择之一，为开发者提供了分布式系统需要的配置管理、服务发现、断路器、智能路由、微代理、控制总线、一次性Token、全局锁、决策竞选、分布式会话与集群状态管理等能力和开发工具。
- Eclipse MicroProfile作为Java微服务开发的基础编程模型，它致力于定义企业Java微服务规范，MicroProfile提供指标、API文档、运行状况检查、容错与分布式跟踪等能力，使用它创建的云原生微服务可以自由地部署在任何地方，包括服务网格架构。
- Tars是腾讯将其内部使用的微服务框架，包含一整套开发框架与管理平台，兼顾多语言、易用性。高性能与服务治理，理念是让开发更聚焦业务逻辑，让运维更高效。
- SOFAShark是由蚂蚁金服开源的一套用于快速构建金融级分布式架构的中间件，也是在金融场景里的最佳实践。
- Dapr（分布式应用运行时）是微软新推出的一种可移植的、无服务器的、事件驱动的运行环境，它使开发人员可以轻松构建弹性，无状态和有状态微服务，这些服务运行在云和边缘上，并包含多种语言和开发框架。



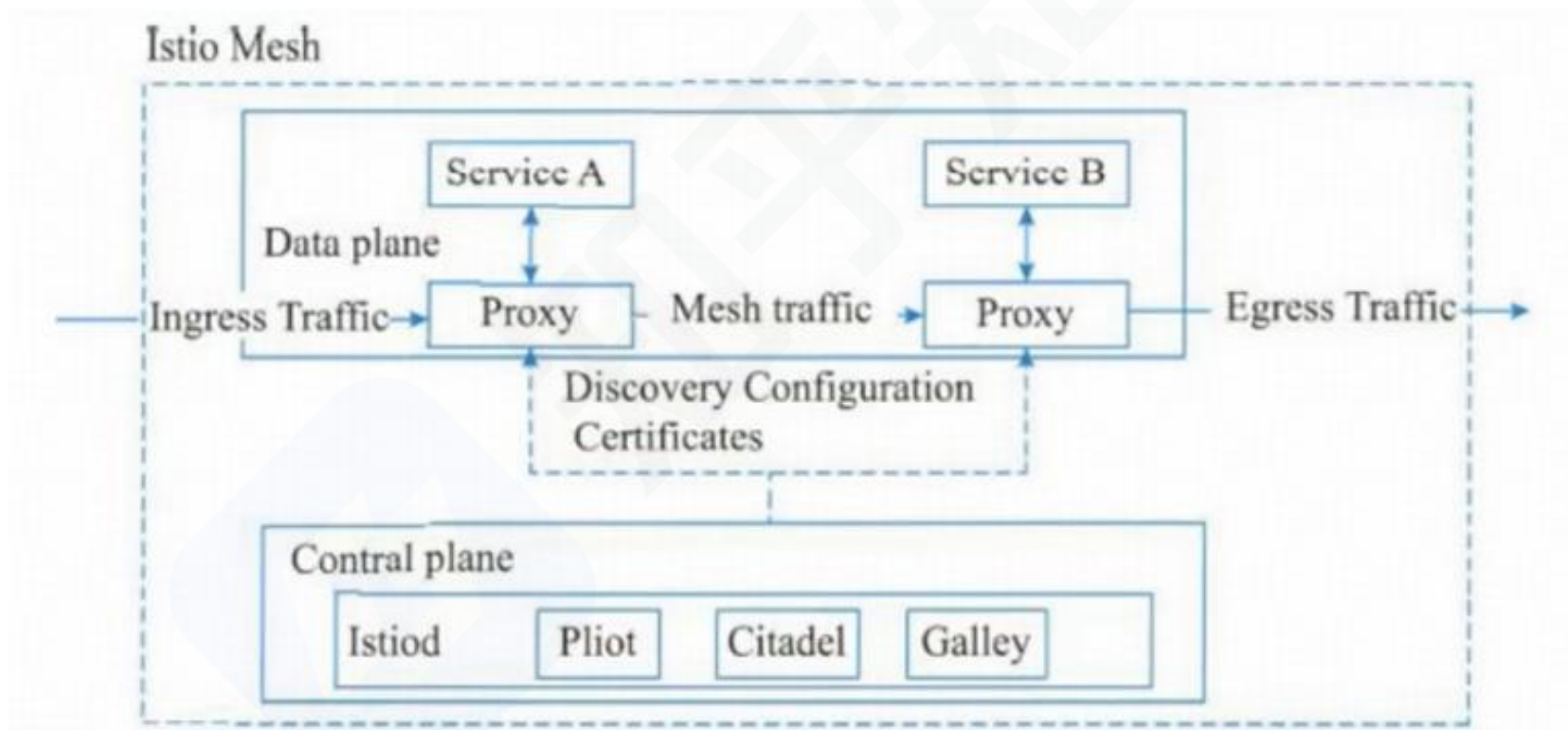
云原生架构相关技术

- **无服务器技术**（Serverless）因为屏蔽了服务器的各种运维复杂度，让开发人员可以将**更多精力用于业务逻辑设计与实现**，而逐渐成为云原生主流技术之一。Serverless计算包含以下特征：
 - **全托管的计算服务**，客户只需要编写代码构建应用，无需关注同质化的、负担繁重的基于服务器等基础设施的开发、运维、安全、高可用等工作；
 - **通用性**，结合云BaaSAPI的能力，能够支撑云上所有重要类型的应用；自动弹性伸缩，让用户无需为资源使用提前进行容量规划；
 - **按量计费**，让企业使用成本得有效降低，无需为闲置资源付费。
- 函数计算（FaaS）是Serverless中最具代表性的产品形态。通过把应用逻辑拆分多个函数，每个函数都通过事件驱动的方式触发执行。
- 无服务器技术关注点：计算资源弹性调度、负载均衡和流控、安全性。



云原生架构相关技术

- **服务网格（ServiceMesh）**是分布式应用在微服务软件架构之上发展起来的新技术，旨在将那些微服务间的连接、安全、流量控制和可观测等通用功能下沉为平台基础设施，实现应用与平台基础设施的解耦。这个解耦意味着**开发者无需关注微服务相关治理问题而聚焦于业务逻辑本身**，提升应用开发效率并加速业务探索和创新。
- 在这张架构图中，服务A调用服务B的所有请求，都被其下的服务代理截获，代理服务A完成到服务B的服务发现、熔断、限流等策略，而这些策略的总控是在控制平面（ControlPlane）上配置。



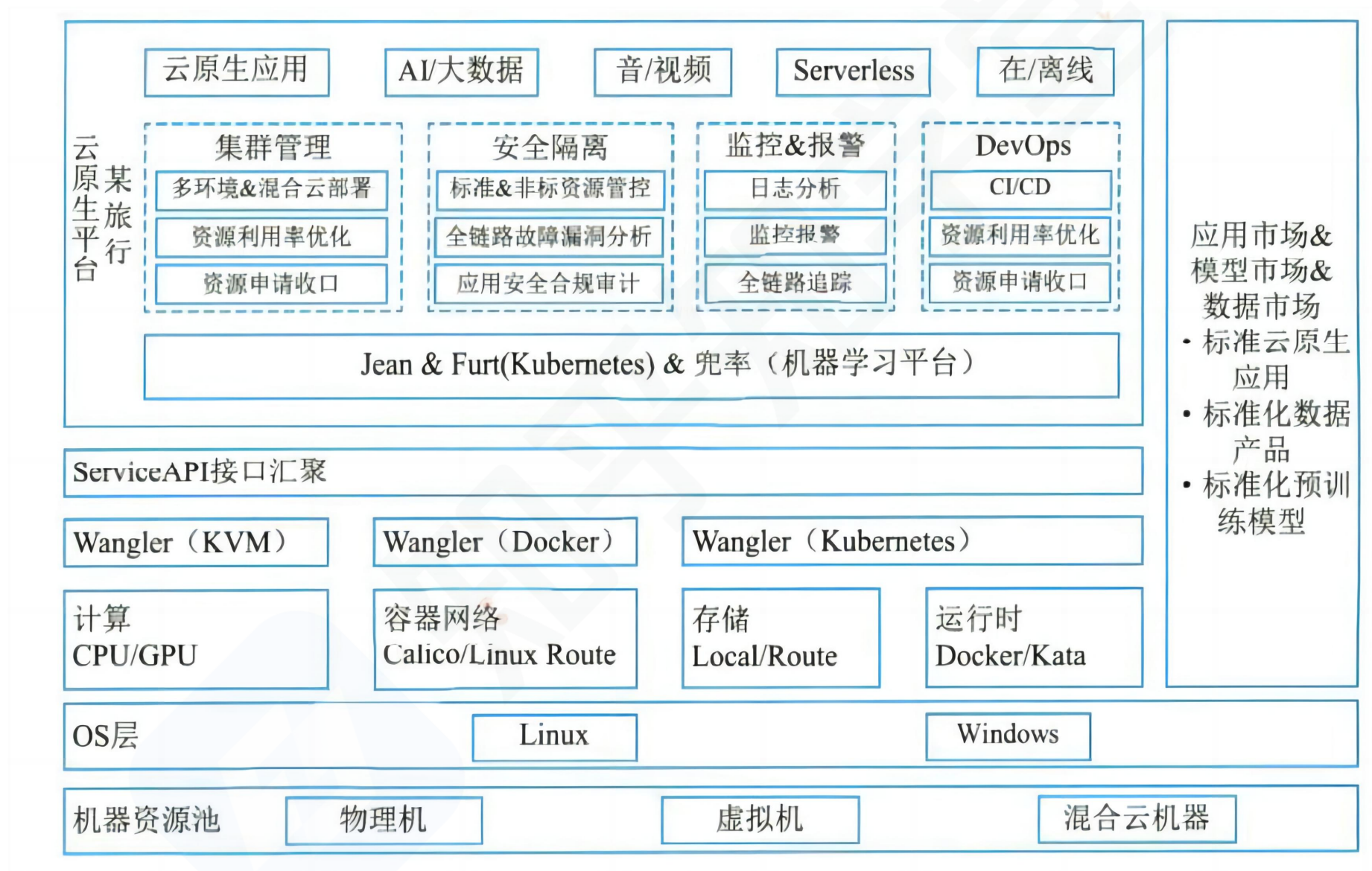


云原生架构案例分析

- **某旅行公司云原生改造：**某公司主体由两个公司合并之后，出现了以下2个问题：**技术体系不同，需要整合为一体；节假日高并发流量。**
- 改造第一阶段，某旅行技术团队为了提升集群资源利用率，降低资源使用成本。利用云原生思维重构部分技术体系，将多套旧有系统合并、收拢到一套以云原生应用为核心的私有云平台上，同时将IDC、物理网络、6虚拟网络、计算资源、存储资源等通过IaaS、PaaS等，实现虚拟化封装、切割再投产的自动化流程。随着服务器集群规模的扩大，部分机器开始频繁出现故障。此时，保障服务稳定性成了第二阶段改造的首要任务。
- 第二阶段基于**公有云、私有云和离线专属云集群**等新型动态计算环境，某旅行公司的技术团队帮助业务构建和运行具有弹性的云原生应用，促进业务团队开始使用声明式API，同时通过不可变基础设施、服务网格和容器服务，来构建容错性好、易于管理和观察的应用系统，并结合平台可靠的自动化恢复、弹性计算来完成整个服务稳定性的提升。
- 第三阶段**通过基础组件、服务的云原生改造、服务依赖梳理和定义等方式**，使应用不再需要考虑底层资源、机房、运行时间和供应商等因素。此外，还利用标准的云原生应用模型，实现了服务的跨地域、跨云自动化灾备、自动部署，并向云原生场景下的DevOps演进。架构图如下：



云原生架构案例分析





云原生架构案例分析

云原生技术助力某汽车公司数字化转型实践：

- 战略性构建容器云平台。通过平台实现对某云行App、二手车、在线支付、优惠券等核心互联网应用承载。以多租户的形式提供弹性计算、数据持久化、应用发布等面向敏捷业务服务，并实现高水平资源隔离。标准化交付部署，快速实现业务扩展，满足弹性要求。
- **数字混合云交付**。采用**私有云+公有云**的混合交付模式，按照服务的敏态/稳态特性和管控要求划分部署，灵活调度公有云资源来满足临时突发或短期高TPS业务支撑的需求。
- **深度融合微服务治理体系**，实现架构的革新和能力的沉淀，逐步形成支撑数字化应用的业务中台



T H E E N D

功不唐捐，玉汝于成！

• 开 启 新 征 程 •

