### 云原生简介

云原生是一系列云计算技术体系和企业管理方法的集合，既包含了实现应用云原生化的方法论，也包含了落地实践的关键技术。云原生应用利用容器、服务网格、微服务、不可变基础设施和声明式 API等代表性技术，来构建容错性好、易于管理和便于观察的松耦合系统，结合可靠的自动化手段可对系统做出频繁、可预测的重大变更，让应用随时处于待发布状态，云原生技术有利于各组织在公有云、私有云和混合云等新型动态环境中，构建和运行可弹性扩展的应用，借助平台的全面自动化能力，跨多云构建微服务，持续交付部署业务生产系统。

云原生方案的重点是如何构建、部署和管理应用。云原生方案与微服务架构类似。然而，尽管微服务可通过构建云原生应用来交付，可企业仍需采取许多措施，才能在生产中熟练地管理微服务。而要想享受云原生应用的各种益处，也并非一定需要微服务。很多企业都通过基于相同的原则，构建出更优秀的模块化单体式应用，从而取得云原生方案的种种效益。

向云原生应用的开发和交付转型，是一次全方位的变革，涉及企业的文化、流程、架构和技术。

在部署方面，虚拟机分钟级的弹性不再满足快速扩缩容的需求，更加轻量级的容器技术成为微服务部署的最佳载体。容器技术很好的解决了应用移植过程的环境一致性问题，使微服务实现快速弹性的部署。敏捷开发带来应用的快速迭代，同时也增加了版本发布的风险与业务运维的复杂度。DevOps 理念提倡开发、测试、运维之间的高度协同，从而在完成高频率部署的同时，提高生产环境的可靠性、稳定性、弹性以及安全性，这在很大程度上消除了频繁发布的风险。

### 云原生架构技术

云原生架构具有如下典型技术特征：采用轻量级的容器、设计为松散耦合的微服务、通过敏捷的 DevOps 流程进行管理。

#### 容器技术

容器其实已有10来年的历史，它具有占用资源少、部署快、易迁移等特点。

云原生应用依靠容器来构建跨技术环境的通用运行模型，并在不同的环境和基础架构（包括公共、私有和混合云）间实现真正的应用可移植性，利用操作系统虚拟化功能来划分多个应用的可用计算资源，并确保这些应用安全无虞、相互独立。云原生应用采用横向扩展模式，因此，只需添加更多应用实例，即可增加容量，而且这样的添加常通过容器基础架构中的自动化功能来实现。由于容器的费用低、密度高，因而可在同一虚拟机或物理服务器内托管大量容器，这使得容器成了交付云原生应用的理想之选。

而在当今市场上，Docker 已成为具有代表性的容器技术，它作为应用级别的容器引擎。具备更加全面的资源控制能力。

结合企业级的场景，面对软件交付，结合数据中心资源等，Docker容器生态技术由容器编排，镜像仓库，容器管理平台等方面构成。

##### 容器编排

应用程序容器的部署和组织是通过容器编排工具完成的。一些流行的开源容器编排工具包括Kubernetes、Docker Swarm和LXC。结合公司技术架构的实际情况，我们选择使用kubernetes作为容器编排工具。kubernetes，简称K8s，它是一个开源的，用于管理云平台中多个主机上的容器化的应用，kubernetes的目标是让部署容器化的应用简单并且高效,Kubernetes提供了应用部署，规划，更新，维护的一种标准化机制。

##### 镜像仓库

容器是由镜像生成，而镜像的获取是从Docker镜像仓库中获取到的，镜像被保存在镜像仓库中，镜像仓库类似于Git中仓库的概念。一个仓库中可以保存很多镜像，或者说很多版本的镜像。也和Git一样，镜像通过标签来标签镜像的不同版本。

docker客户端通过pull和push来下载或上传镜像到仓库。仓库可以是公共的或者私有的，而我们最常使用的共有仓库是Docker Hub。同时，我们还选择搭建Nexus服务作为公司内部使用的私有仓库。

Nexus作为一个强大的仓库管理器,它极大的简化了本地内部仓库的维护和外部仓库的访问，是一套开箱即用的系统不需要数据库,使用文件系统加Lucene来组织数据，它使用ExtJS来开发界面,利用Restlet来提供完整的REST APIs,通过IDEA和Eclipse集成使用，支持webDAV与LDAP安全身份认证。

##### 容器管理平台

容器管理平台针对集群提供的一个可视化界面而进行相关操作和管理，可是使DevOps团队可以轻松测试，部署和管理他们的应用程序。Rancher作为开源软件，可以结合组织需要采用的所有组件，并在生产环境中运行容器。它为运行容器化的应用实现了一层灵活的基础设施服务，包括网络、存储、负载均衡、DNS和安全模块等等，并且，这些基础设施服务也是通过容器部署。我们选择使用rancher作为容器管理平台。

#### 微服务

微服务是一种架构模式，它提倡将单一应用程序划分成一组小的服务，服务之间互相协调、互相配合，为用户提供最终价值。每个服务运行在其独立的进程中，服务与服务间采用轻量级的通信机制互相沟通（通常是基于HTTP的RESTful API）。以往企业应用主要是面向服务的架构（SOA），SOA 是一种粗粒度、松耦合服务架构，服务之间通过简单、精确定义接口进行通讯，不涉及底层编程接口和通讯模型。

微服务框架包括多编程语言支持、服务通信、服务注册、服务发现、认证鉴权、流量控制、熔断隔离、服务容错、服务路由、负载均衡、健康检查、故障注入等功能。

##### 服务网格

服务网格是用于处理服务间通信的专用基础设施层。它负责通过包含现代云原生应用程序的复杂服务拓扑来可靠地传递请求。实际上，服务网格通常通过一组轻量级网络代理来实现，这些代理与应用程序代码一起部署，而不需要感知应用程序本身。

服务网格也被称为Service Mesh，它的功能包括应用程序间通讯的中间层、轻量级网络代理、应用程序无感知、解耦应用程序的重试/超时、监控、追踪和服务发现等等

服务网格的开源软件有 Istio、Linkderd、Envoy、SOFAMesh、Dubbo Mesh 等。

我们选用Istio作为微服务的服务网格，Istio 提供一种简单的方式来为已部署的服务建立网络，该网络具有负载均衡、服务间认证、监控等功能，而不需要对服务的代码做任何改动。它能够实现智能控制服务之间的调用流量，能够实现灰度升级、AB 测试和红黑部署等功能，自动为服务之间的调用提供认证、授权和加密，应用用户定义的 policy，保证资源在消费者中公平分配。，查看服务运行期间的各种数据，比如日志、监控和 tracing，了解服务的运行情况等等。

##### API网关

API网关是一个服务器，是系统的唯一入口。从面向对象设计的角度看，它与外观模式类似。API网关封装了系统内部架构，为每个客户端提供一个定制的API。它可能还具有其它职责，如身份验证、监控、负载均衡、缓存、请求分片与管理、静态响应处理。

API网关方式的核心要点是，所有的客户端和消费端都通过统一的网关接入微服务，在网关层处理所有的非业务功能。通常，网关也是提供REST/HTTP的访问API。服务端通过API-GW注册和管理服务。

我们选择了Kong来作为微服务的API网关，Kong 是在客户端和（微）服务间转发API通信的API网关，通过插件扩展功能，这些插件在 API 请求响应循环的生命周期中被执行。。它主要有两个组件组成，Kong Server和Apache Cassandra。Kong Server是基于nginx的服务器，用来接收 API 请求。而Apache Cassandra用来存储操作数据。Kong具有如下几个基础功能：HTTP 基本认证、密钥认证、CORS（ Cross-origin Resource Sharing，跨域资源共享）、TCP、UDP、文件日志、API 请求限流、请求转发以及 nginx 监控。

#### DevOps

DevOps是云原生不可或缺的一部分，它可以作为过程、方法与系统的统称，用于促进开发、技术运营和质量保障部门等之间的沟通、协作与整合。它的出现是由于软件行业日益清晰地认识到，为了按时交付软件产品和服务，开发和运营工作必须紧密合作。

DevOps 旨在统一软件开发和软件操作，与业务目标紧密结合，在软件构建、集成、测试、发布到部署和基础设施管理中大力提倡自动化和监控。它的目标是缩短开发周期，增加部署频率，更可靠的发布。用户可通过完整的工具链，深度集成代码仓库、制品仓库、项目管理、自动化测试等类别中的主流工具，实现零成本迁移，快速实践DevOps。

要完成云原生应用的构建之路，开发和 IT 运维团队必须进行多方面的变革，以便更加快速高效地构建和部署应用。无论身处哪个行业、规模如何，所有企业都需要周全地考虑各种活动、技术、团队和流程，因为这些要素综合起来才能实现 DevOps 文化。要想充分利用新技术，采用更加快速的方案，实现更为密切的合作，企业必须切实遵循 DevOps 的原则和文化价值，并围绕这些价值来进行组织和规划。

##### DevOps平台

DevOps 平台可以提供项目管理、原型设计、源代码版本管理、代码质量分析、持续交付流水线、编译构建、自动化测试、部署、发布、运维等功能特性。

根据公司组织架构、文化与行为等与 DevOps 契合度方面的问题。我们融合敏捷、精益、自治团队、分布式决策等理念，使用Gitlab作为代码仓库，选择Racnher Pipeline作为持续集成/持续交付的工具。

GitLab 是一个用于仓库管理系统的开源项目，使用Git作为代码管理工具，提供用户界面。并且免费提供无限数量的私人和公共存储库。

市场中大部分的CI/CD工具无法做到同时支持多种源代码管理，甚至暂不支持任何私有仓库。而在Rancher Pipeline中，Rancher创造性地让同一个Rancher用户可以同时使用GitHub与GitLab进行基于OAuth的身份验证，无需插件，即可在单一环境中同时拉取、使用和管理托管在GitHub和GitLab的代码。无缝贴合我们的代码仓库类型。而且Rancher Pipeline Pipeline的用户界面与操作体验秉承了Rancher容器管理平台一贯广为用户所喜爱的简洁、友好的优点，将用户从繁琐复杂的代码与命令行中解放出来，一切pipeline配置均已可视化，用户可以轻松快速地以拖拽方式来构建pipeline