

# 基于 Kubernetes 的融合云原生基础设施方案与关键技术

何震苇，黄丹池，严丽云，林园致，杨新章
（中国电信股份有限公司研究院，广东 广州 510630）

**摘　要：**随着云原生技术在 IT 域和 CT 域的广泛应用，云原生基础设施逐渐成为新一代的云计算基础设施。分析了主流云原生基础设施技术体系及其局限性，提出了基于 Kubernetes 的轻量级融合云原生基础设施及其管理方案，探讨了融合云原生基础设施的关键技术和应用场景。

**关键词：**云原生基础设施；云原生；容器；Kubernetes；KoK
**中图分类号：**TP392
**文献标识码：**A
**doi:** 10.11959/j.issn.1000−0801.2020314

# Kubernetes based converged cloud native infrastructure solution and key technologies

HE Zhenwei, HUANG Danchi, YAN Liyun, LIN Yuanzhi, YANG Xinzhang
Research Institute of China Telecom Co., Ltd., Guangzhou 510630, China

**Abstract:** With a wide range of applications in IT domain and CT domain of cloud native technology, cloud native infrastructure is becoming the next generation of cloud infrastructure. The mainstream technology system and its limitations of cloud native infrastructure were analyzed, a lightweight converged cloud native infrastructure and its management scheme based on kubernetes was proposed, and the key technologies and application scenarios of converged cloud native infrastructure were discussed.

**Key words:** cloud native infrastructure, cloud native, container, Kubernetes, KoK

## 1　引言

随着 5G、边缘云、虚拟专网、AI 等新业务的兴起以及 CT 与 IT 技术加速融合发展，运营商尝试引入在 IT/互联网业界得到广泛应用的云原生基础设施，承载 5GC、MEC 等云原生网络业务，提供相对传统 IaaS 云资源更敏捷灵活的业务交付能力、更弹性智能的运维管理能力和更节能高效的资源利用能力。在云原生计算基金会（CNCF）和主流互联网云服务商的推动下，云原生基础设施已经形成了跨越 IaaS 和 PaaS 的庞大开源技术体系，出现了多种云原生基础设施开源组件和技

收稿日期：2020−11−01；修回日期：2020−12−07
基金项目：国家重点研发计划基金资助项目（No.2018YFB1802400）
**Foundation Item:** The National Key Research and Development Program of China (No.2018YFB1802400)

术方案。

针对 5G 时代网络云大带宽低时延的性能需求、边缘协同的资源需求和多厂商多行业承载的安全需求，本文分析了云原生基础设施技术体系及其主要技术方案，结合云原生技术发展趋势和网络云需求重点探讨了基于 Kubernetes（以下简称 K8S）的融合云原生基础设施关键技术和管理方案，为运营商云原生基础设施演进提供指引。

## 2 云原生基础设施技术体系

云原生基础设施是承载云原生应用的软硬件资源集合，容器是云原生基础设施的核心技术，由于容器技术本身兼具资源属性和应用属性，因此云原生基础设施也跨越了传统的 IaaS 和 PaaS 边界。在 IaaS 层面云原生基础设施包括基于容器技术体系构建的云原生计算、云原生存储和云原生网络资源，而在 PaaS 层面，云原生基础设施又包括基于容器及其编排能力构建的各类云原生 PaaS 服务，包括云原生 CI/CD、云原生数据库、服务网格、无服务器服务、云原生 AI 服务等能力和服务。云原生基础设施技术体系如图 1 所示。

在云原生基础设施中，容器服务（container as a service）是整个基础设施的核心，提供容器编排、部署、运行和调度能力，而 K8S 已成为容器服务的事实标准。K8S 由控制面和工作面构成，控制面提供基于声明式 API 的容器编排调度能力，工作面提供基于 Pod 的容器实例的部署运行能力，K8S 本身并没有提供完整的存储、网络和安全能力，需要通过某种方式集成第三方存储、网络和安全组件才能提供完整的基础设施能力。

目前构建 K8S 使能的云原生基础设施主要有 3 种技术方案：裸机 K8S 和 KoO（K8S over OpenStack）、KoK（K8S over K8S）。

（1）裸机 K8S 云原生基础设施

裸机 K8S 模式直接在物理服务器上部署 K8S 集群，K8S 集群的控制面和工作面均采用裸机部署，工作面直接通过 CSI 对接 Ceph/IPSAN 等物理存储集群提供容器存储服务，直接在物理网络上构建基于 CNI 的扁平化全联通容器网络，如 Calico、Flannel 等。裸机 K8S 模式只能运行裸机容器，裸机容器直接在物理机操作系统上运行，具有很高的通用计算性能，而且裸机部署的集群控制面也能支持很大的容器集群规模，而由于裸机 K8S 直接依赖 K8S 和容器自身的安全机制，其多租户安全隔离能力较弱，同时集群控制面和存储资源都基于独立物理机部署，控制面和存储面的资源利用率较低，此外直接基于物理机提供容器服务也意味着容器服务和上层的云原生应用与底层的物理硬件架构强绑定。裸机 K8S 基础设施比较适合互联网巨头承载企业内部的大型纯容器云原生应用，能够最大限度地发挥大规模物理服务器集群的性能和资源规模调度的效率优势。裸机 K8S 云原生基础设施如图 2 所示。

（2）KoO 云原生基础设施

KoO 模式将 K8S 部署在 OpenStack IaaS 云资源池之上，由 OpenStack 对托管的 K8S 集群进行

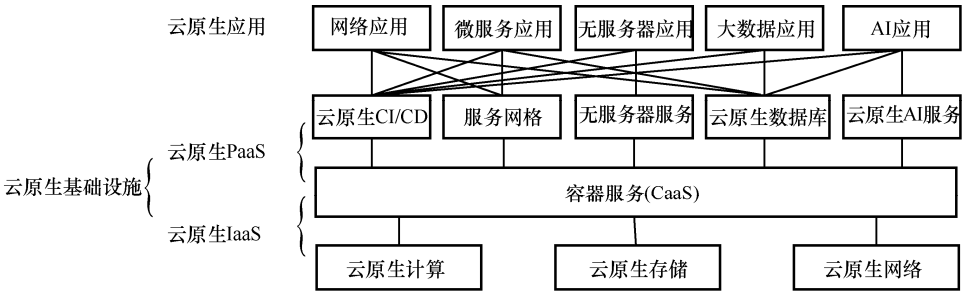



图 1 云原生基础设施技术体系


图 2　裸机 K8S 云原生基础设施

全生命周期管理，K8S 集群的控制面和工作面均采用 OpenStack 资源池的虚拟机部署，集群的存储面和网络面也基于 OpenStack 资源池提供的虚拟机存储和虚拟网络资源构建。基于 OpenStack 搭建 K8S 基础设施的好处是能够充分利用成熟的 OpenStack IaaS 云资源快速搭建云原生基础设施，同时借助虚拟机和 IaaS 现有的安全机制提供较为完善的租户安全隔离能力。

由于目前主流的运营商大多构建了基于 OpenStack 的 IaaS 云资源池，在 OpenStack 之上部署 K8S 也成为诸多运营商的自然选择。然而 KoO 方式也有明显的缺点：首先，在于将容器和容器网络都运行在虚拟化层之上，对通用容器计算和网络的性能有明显的损耗；其次，OpenStack 本身是一套庞大的软件体系，涉及诸多 Python 组件，不太适合在资源受限的边缘云、边缘站点上部署；再次，K8S 是容器编排的事实标准，而 OpenStack 本身也提供了一些容器服务（如 Zun），这些容器服务无法通过 K8S 进行统一编排调度；最后，OpenStack 本身的软件架构和语言实现效率较低，IaaS 层资源服务开通速度较慢，以 OpenStack 官方推荐的 Kuryr+Neutron 网络方案为例，为 K8S 的容器创建网络端口，涉及 KuryrCNI 插件、NeutronServer、Keystone、NeutronDB、MQ、Agent、OVS 等多个组件的调用，整个调用链很长，使得创建网络端口甚至比创建容器本身更花时间，难以满足大规模云原生基础设施的资源快速开通需求。KoO 云原生基础设施如图 3 所示。

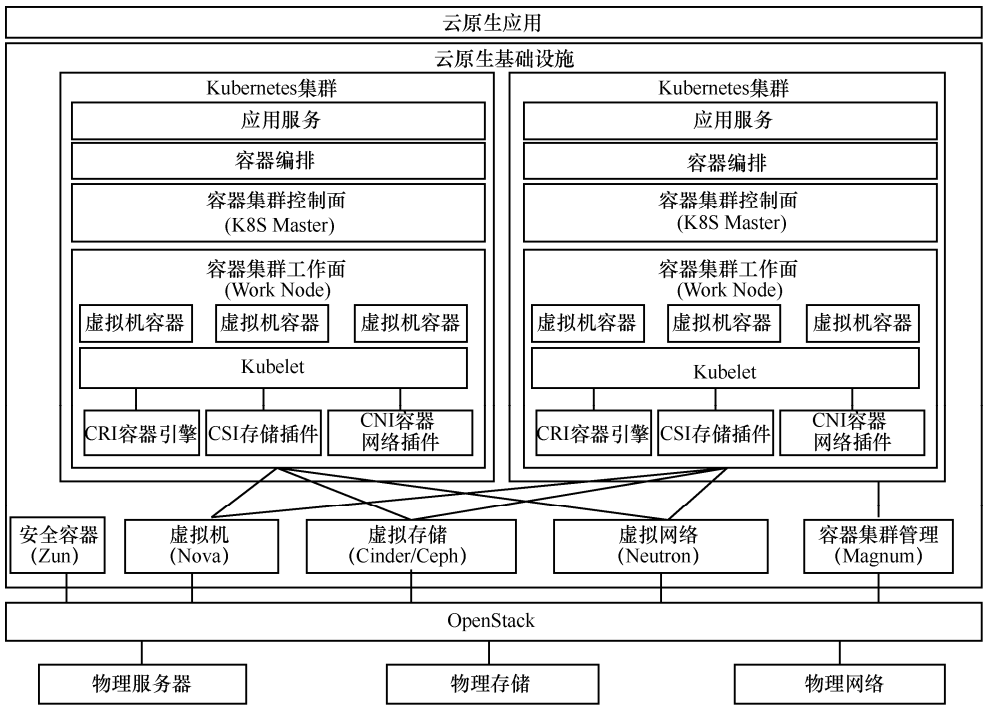



图 3　KoO 云原生基础设施

（3）KoK 云原生基础设施

KoK 模式先在物理机之上构建基础 K8S 集群（也称主集群），在基础 K8S 集群之上以云原生方式构建基础的计算、存储和网络资源，然后再基于这些基础云原生资源构建客户 K8S 集群，客户集群的控制面可以采用基础集群提供的物理机、虚拟机或容器构建，客户集群的工作面可以采用物理机或虚拟机构建。客户集群的工作面直接对接基础集群提供的云原生存储和网络资源。每个客户集群拥有自己的容器编排和应用服务能力，而在基础集群层面，可以提供跨集群的全局容器编排和应用服务能力。这种模式兼具裸机 K8S 高性能、高效率、扁平化的优点，又具备 KoO 集群兼容性好、隔离度高的特点，是极具潜力的一种云原生基础设施技术路线。KoK 模式的主要问题在于 K8S 官方并没有提供虚拟机管理、存储管理、统一组网和租户安全能力，这些能力需要依赖和集成第三方开源组件，目前 CNCF 社区相关领域的开源组件众多，成熟度不一，需要经过严格的评估、选型、测试甚至优化定制才能满足网络云高性能、高可靠、高安全的独特要求。KoK 云原生基础设施如图 4 所示。

从上述分析可以看出，K8S、KoO、KoK 这 3 种主流的云原生基础设施构建方式均有其优势和短板。裸机 K8S 的优势是高性能、运维简单，短板是安全隔离能力弱，传统应用兼容性差；KoO 的优势是安全、成熟，能兼容现有虚拟化承载业务，短板是性能较差，边缘部署难度大；而 KoK 模式兼具前两者的优点，其短板是在云原生存储、网络和安全能力成熟度上较

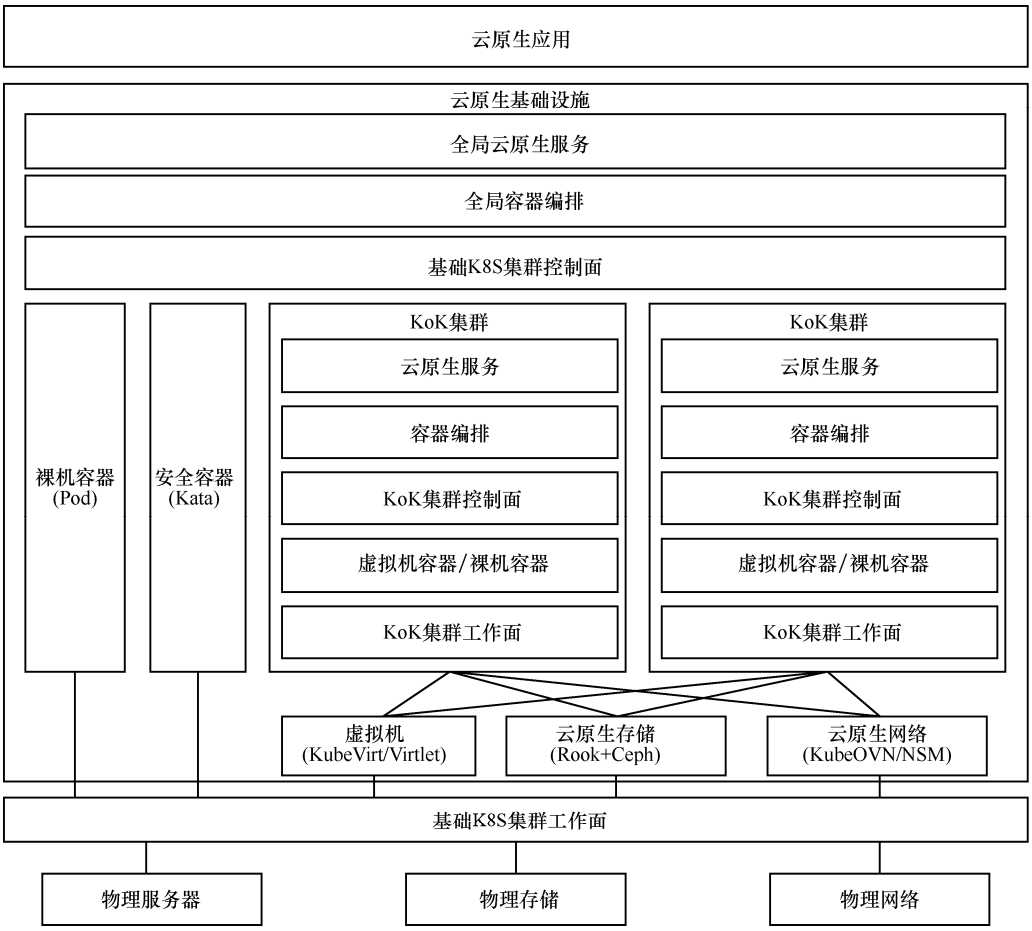



图 4　KoK 云原生基础设施

弱，本文将重点分析基于 KoK 的融合基础设施及其管理方案。

## 3　融合云原生基础设施及其管理方案

融合云原生基础设施整体方案直接基于物理机资源池构建，由融合云原生基础设施（cloud native infrastructure，CNI）和融合云原生基础设施管理平台（cloud native infrastructure management，CNIM）构成，如图 5 所示。融合基础设施包括物理机池、基础 K8S 集群、由基础 K8S 集群承载的云原生计算、云原生存储和云原生网络资源，其中云原生计算资源包括裸机容器、KVM 虚拟机、Kata 安全容器和 KoK 客户集群，云原生存储资源包括容器化部署的 Ceph 存储集群及其存储管理器，云原生网络资源包括容器化部署的网络控制面和网络转发面，以及与网络转发面集成的网络加速组件。

融合基础设施管理平台由基础设施交付、资源管理公共服务、基础资源管理、云原生资源服务管理、API 服务总线、统一安全管理和统一监控维护等部件构成，构建在基础 K8S 集群控制面之上作为管理多个基础 K8S 集群的全局资源控制面，对融合云原生基础设施的各类资源和服务进行统一交付、编排、调度和管理，通过 API 总线以声明式 API 的形式提供各类资源服务的管理能力，同时通过 API 总线代理主集群和 KoK 客户集群的控制面接口，提供标准 K8S API，完全兼容 CNCF 社区的各类基于 K8S API 的工具链，此外通过 API 总线还可以提供与 NFVO 和 NFVM 对接的接口，将云原生基础设施融入 NFV 标准化技术体系。

### 3.1　物理主机纳管方案

基于物理机构建融合云原生基础设施，首先要实现物理机的池化管理。物理机的池化管理涉及服务器架构、服务器硬件、主机操作系统、主机基础组件和主机网络的管理。在服务器架构层面需涉及 x86 架构和 ARM 架构，x86 架构有生态和成熟度优势，适合承载通用、核心业务，ARM 架构有节能优势，适合承载边缘业务，未来还可支持自主可控的 RISC-V 架构。在服务器硬件层面主要涉及计算、内存、磁盘和网络等设备的管理，其中计算设备包括 x86 CPU、ARM CPU、计

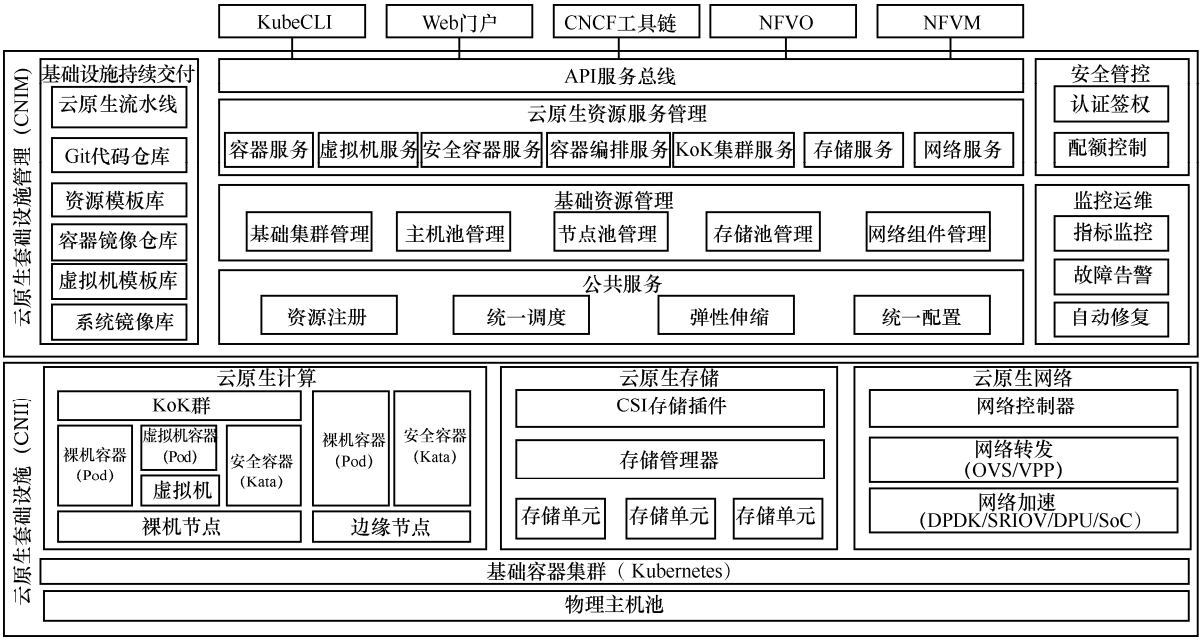



图 5　融合云原生基础设施整体方案

算加速的 GPU/FPGA，网络云对 I/O 性能和网络性能的要求较高，磁盘设备一般采用高 I/O 吞吐的 SSD 存储，网络设备通常会采用 25 GB 以上的通用网卡和 100~200 GB 具备 DPU/FPGA/SoC 等网络加速和网络处理卸载能力芯片的智能网卡。在操作系统层面，云原生基础设施的物理机除了采用 CentOS 等通用操作系统之外，还可以采用 CoreOS/RancherOS 等容器云原生操作系统，云原生操作系统通常内置了容器引擎，针对容器应用场景进行了系统内核裁剪和性能、安全上的优化，比通用 OS 体积更小、更加安全，也更易升级。在基础组件层面，需要部署用于纳管物理机的容器引擎、监控代理和网络组件，监控代理和网络组件均可采用容器方式部署。

物理机的资源配置信息保存在基础集群的控制面，并且通过物理机资源控制系统管理物理机生命周期。物理机操作系统安装采用 PXE 自举方式安装，在开机时首先连接数据中心的 PXE 服务器，下载 PXE 引导系统，PXE 引导系统先从基础集群控制面获取物理机基础配置信息，基于这些信息从系统镜像仓库选择合适的操作系统镜像进行安装，物理机资源控制器检测到物理机 OS 安装成功之后，会采用 Ansible 安装标准容器引擎（包括 Kubelet、ContainerD、RunC 等），并通过基础集群容器接口安装监控组件、网络组件（如 OVS、DPDK 等）等基础组件，最终将物理机作为基础资源纳管到基础集群中。物理机资源控制器实时监控物理机状态，并通过声明式 API 控制物理机启动、关闭、重启、卸载、重装等全生命周期。

### 3.2 虚拟机纳管方案

CNCF 社区目前主要有两种基于 K8S 的虚拟机管理方案：KubeVirt 和 Virtlet，两者均提供北向 CRI，南向对接 libvirt，都能提供基本的虚拟机开关启停操作，相比较而言，KubeVirt 提供了独立的虚拟机资源控制器，能够实现接近 OpenStack 的虚拟机全生命周期管理能力，而 Virtlet 主要依赖 K8S 现有的 Pod 级别资源控制器，好处是能够无缝对接 K8S 的容器编排体系，可以像编排 Pod 那样编排虚拟机。由于在云原生基础设施上引入虚拟机主要是为了兼容现有的虚拟化网元应用承载，对虚拟机的使用和管理更接近传统虚拟化体系，而且虚拟机本身较重，不太适合采用 Pod 容器那种快速重启调度的管理方式，因此 KubeVirt 更适合用于融合云基础设施对虚拟机的管理。

### 3.3 容器纳管方案

在融合云基础设施中可以提供裸机容器、虚拟机容器和安全容器资源，裸机容器直接在运行基础集群或 KoK 集群的裸机工作节点上，具有最佳的综合性能和最快的调度速度，虚拟机容器运行在 KoK 集群的虚拟机工作节点上，能利用虚拟机的安全隔离能力实现较佳安全性，安全容器运行在轻量级虚拟机中，通过轻量虚拟机实现容器安全隔离，既有类似裸机容器的快速启停能力，也有和虚拟机一样的安全能力。这 3 种容器均由支持 CRI 北向接口的容器引擎驱动（如 ContainerD）运行，都能通过 K8S 现有容器编排机制进行编排调度。

对于裸机容器和虚拟机容器，目前业界通常使用 K8S 内置的 Docker 引擎进行管理，由于 Docker 本身集成了很多 K8S 不会用到的功能，如内置 Swarm，而且 Docker 本身受美国 Docker 公司严格控制，未来升级使用可能受限，更标准合理的做法是部署北向支持 CRI 的容器引擎 ContainerD 和南向支持 OCI 的标准化容器引擎 RunC 代替 Docker，管理裸机容器和虚拟机容器。

对于安全容器，目前最成熟的安全容器引擎是 Kata，Kata 本身支持 OCI，能够与 ContainerD 容器引擎集成，而 Kata 容器所依赖的轻量级虚拟机镜像也可以由基础集群的虚拟机镜像仓库统一管理。

此外，借助 K8S 社区的 RuntimeClass 机制，

在容器的声明式 API 中添加 RuntimeClass 参数，指定 Pod 关联的容器引擎，可以实现多个异构容器引擎资源统一调度。

### 3.4　基础 K8S 集群构建方案

K8S 基础集群是整个融合云原生基础设施的核心资源，基于底层物理机资源池构建，基础集群部署涉及集群的控制面、工作面以及集群级的网络组件和存储组件的部署。

控制面部署主要是 K8S Master 节点相关组件的部署，包括 APIServer、Scheduler、ControllerManager、ETCD 等组件，这些组件需要按照它们的依赖关系依次安装，并且一般需部署奇数个 Master 控制节点以保证整个基础集群控制面的高可用。可以将基础集群控制面的部署过程定义成 Ansible 部署模板，包括需要部署的集群控制面组件、组件的安装次序、依赖的物理机规格、默认部署节点数、默认主节点等信息，通过 Ansible 脚本引擎选择合适的物理机完成控制面自动部署。

工作面部署主要是 K8S 工作节点相关组件的部署，包括 Kubelet、KubeProxy 和容器引擎。由于 Kubelet 和 KubeProxy 的每个工作节点都需安装，而安装的版本相同，这两个组件的安装可以前置到物理机节点纳管过程中，作为物理机的基础组件安装，而容器引擎既有 ContainerD+RunC 标准容器引擎，又有 KubeVirt、Kata 等扩展容器引擎，标准容器引擎通常是每个工作节点都需要部署的，也可以前置作为物理机基础组件安装，而 KubeVirt、Kata 容器则采用 Ansible 模板按需部署。

网络组件的部署通常涉及网络控制组件、网络 CNI 插件、网络转发组件和网络加速组件的部署，目前 CNCF 社区开源的 KubeOVN、Calico 等 K8S 网络组件均支持容器方式安装。网络控制组件可以以 DaemonSet 容器方式部署在控制面节点中，网络 CNI 插件和网络转发组件可以以 DaemonSet 容器方式部署在基础集群的所有控制面和工作面节点上，而网络加速组件通常与服务器的硬件配置相关，需要结合服务器配置标签部署在合适的工作面节点上。

对于存储组件，目前社区最成熟的云原生存储方案是 Rook+Ceph，RooK 定义了基于 K8S 的 Ceph 的容器化部署模板和存储资源控制器，通过 Rook 能够将 Ceph 的 Manager、Monitor、OSD 等部件以容器方式部署到 K8S 集群的工作节点中，实现云原生计算资源和存储资源融合部署。

在资源充足的核心 DC，基础集群的控制面和工作面可独立部署在不同的物理机上，而在资源受限的边缘 DC 或边缘站点，基础集群的控制面和工作面可以部署在相同的物理机上。

### 3.5　KoK 集群构建方案

KoK 集群部署和运行在云原生基础设施的基础 K8S 集群工作面之上。KoK 集群部署同样涉及集群控制面和集群工作面的部署，KoK 集群的控制面可以部署在基础集群的虚拟机、裸机容器或安全容器上，而工作面可以部署在基础集群的裸机和虚拟机上。针对控制面和工作面的不同部署类型，可以提供虚拟机−裸机、虚拟机−虚拟机、容器−裸机、容器−虚拟机等多种 KoK 容器集群模板。由于 KoK 容器集群所有依赖的组件和资源均由基础 K8S 集群管理，KoK 集群的部署可以充分利用基础集群提供的容器编排机制、云原生流水线以及 Helm、Operator 等云原生模板和控制器机制实现灵活高效的自动化部署和自动化运维。

为了实现更高效的资源利用，还可以针对具体的部署模板和部署流程进行优化。例如，对于容器部署的控制面，由于 ETCD 组件的性能、可靠性和资源要求较高，可以采用多集群共享 ETCD 方案，通过在 ETCD 数据路径中引入集群标识和访问控制机制实现多集群控制面数据的统一存储和隔离。对于物理机部署的工作节点，由于工作节点上部署的 Kubelet 已经接入基础集群控制面，需要在工作节点上再部署一套接入 KoK 集群控制

面的 Kubelet，并在基础集群中将该节点设为不可调度。

## 4 融合云原生基础设施关键技术

### 4.1 自定义资源扩展

融合云原生基础设施需要引入和管理大量 K8S 本身没有的资源，基于 K8S 的开放式架构和资源扩展机制，可以对硬件设备、容器运行时、存储能力、网络能力、容器编排能力进行扩展。

- 硬件设备扩展：K8S 原生只支持 x86 和 ARM CPU，通过 Device Plugin API 可以实现 Kubelet 对 GPU、DPU、NP、FPGA 等计算和网络加速硬件设备的注册、管理和调度。
- 容器运行时扩展：通过 CRI， Kubelet 可以对接各类支持 CRI 的容器运行时或虚拟机引擎，包括 ContainerD、CRI-O、Kata（安全容器）、Kubevirt（虚拟机）、Virtlet（虚拟机）等，实现标准 OCI 容器、安全容器和虚拟机的统一生命周期管理。
- SideCar 容器扩展：SideCar 容器是 Kubelet 在启动 Pod 时动态注入的旁挂容器，通常在主容器运行之前启动，在主容器关闭之后关闭，可以在 SideCar 容器中扩展主容器的操作原语，例如增加容器在线迁移能力、设备重定向能力等，或者提供类似 ServiceMesh 或 Network ServiceMesh（NSM）的容器流量控制和转发能力。
- 存储能力扩展： Kubelet 可以接入各类适配 CSI 的分布式存储服务，包括 Ceph、Glusterfs 等，同时分布式存储服务的相关组件也可以容器化部署在 K8S 上，成为基于 K8S 的云原生存储。
- 网络能力扩展：Kubernetes 的网络能力可通过 CNI 插件扩展，由于 CNI 本身定义得过于简单，仅支持单网卡单 IP 扁平网络模型，多网卡绑定、子网绑定、网络 QoS、VPN 等高级网络功能需要通过引入网络控制器甚至网络 SideCar 容器实现。
- 容器编排能力扩展：Pod 是容器工作负载的最小调度单元，K8S 内置了若干基于 Pod 的标准工作负载，如 Deployment、StatefulSet、Job、DaemonSet 等。可通过 K8S 提供的 CRD 机制定义新的资源对象，并采用 Operator 模式实现新资源对象的资源控制器，将资源控制器以容器插件形式部署到 K8S 控制面，实现新资源对象的生命周期自主控制。通过 CRD 和 Operator 机制可以实现物理机、KoK 集群、虚拟机、安全容器等扩展资源的 API 定义和生命周期管理。

### 4.2 自动化弹性伸/缩

构建融合云原生基础设施的弹性伸/缩能力需要借助 K8S 自身的弹性伸/缩机制，并进行适当的扩展。K8S 提供了面向 Pod 的水平伸/缩（HPA）和垂直伸/缩（VPA）机制，水平伸/缩适用于无状态容器，可以通过容器的负载动态增加或减少容器实例的数目，垂直伸/缩适用于有状态容器，可以动态增加或减少指定的容器实例资源规格。由于 K8S 的原生 Kubelet 不支持在线修改容器配置，对 Pod 垂直伸缩需要在主机上重启 Pod。

超融合云原生计算资源的裸机容器、虚拟机容器、虚拟机、安全容器可以沿用 K8S 现有的水平伸/缩机制实现，而由于 K8S 现有垂直伸缩机制需要重启 Pod，可能造成部分业务繁忙的有状态容器业务中断，因此需要扩展 Kubelet 的 UpdatePod 接口，支持有状态容器、虚拟机、安全容器的内存资源在线变更。

对于托管的 K8S 集群，集群资源弹性伸/缩涉及集群控制面和工作面的弹性伸/缩。控制面弹性伸/缩通常以 K8S Master 节点为伸/缩单元进行水平伸/缩或垂直伸/缩，由于 K8S Master 依赖的 Raft

算法在选举 Master 节点时需要过半数节点投票，因此控制面节点只能以奇数个节点为目标进行弹性伸/缩。工作面的伸/缩主要是对 K8S 工作节点进行水平或垂直伸/缩，工作节点伸/缩的下限不能小于已分配资源，上限不能超出控制面管理的容量限制。工作节点回收需要驱逐节点上的容器，无状态容器可直接驱逐，有状态容器需借助 Operator 或相关自动化工具配合迁移。

### 4.3　多层次统一安全机制

融合的云原生基础设施的安全机制涉及控制面、运行面、网络面和数据面的安全机制。控制面安全涉及全局控制面（云原生基础设施管理平台）、基础集群控制面和 KoK 集群控制面的安全管控，而运行面安全涉及工作节点和容器工作负载的安全隔离，网络面安全主要涉及集群网络和租户网络隔离，而数据面安全涉及镜像、模板等基础设施安装软件以及系统和租户的配置数据、运行数据等软件和数据加密、鉴权和防篡改。本节重点分析控制面安全机制。

在控制面安全上，为了兼顾认证鉴权的可控性和高效性，采用统一认证、分布式鉴权的思路。在全局控制面提供统一的认证和鉴权机制，通过开源 LDAP 数据库统一管理用户数据库，通过开源 OIDC（OpenID connect）提供统一认证服务，通过开源 RBAC 框架 Casbin 提供全局 RBAC 鉴权能力。全局控制面管理全局资源鉴权策略、基础集群鉴权策略和 KoK 集群鉴权策略，将基础集群鉴权策略下发给基础集群控制面，将 KoK 集群鉴权策略下发给 KoK 集群控制面，并且对下发的鉴权策略进行监听，自动覆盖不一致的鉴权策略。全局资源鉴权由全局控制面执行，基础集群资源鉴权由集群集群控制面执行，而 KoK 集群级资源的鉴权由 KoK 集群控制面执行。多层级资源 RBAC 鉴权模型如图 6 所示。

### 4.4　多租户集群网络

基础集群中的多个 KoK 租户集群采用基于 KubeOVN 的统一组网方案，通过 KubeOVN 的 CNI 插件和网络控制器对接 OVN 北向接口，通过 OVN 对接 OVS。基于 OVN 路由器实现支持三层转发的 vRouter，基于 OVS 实现 vSwitch 二层 overlay 网络转发。

在基础集群部署一套全局 vRouter 转发跨集

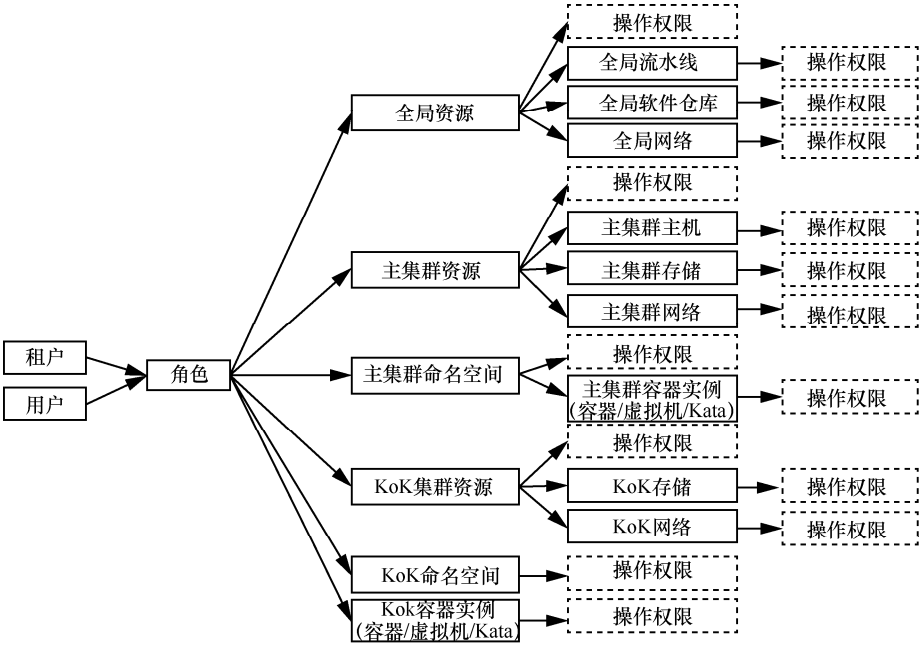



图 6　多层级资源 RBAC 鉴权模型

群、跨租户的全局流量，在 KoK 集群部署一套集群 vRouter，转发 KoK 集群内跨租户的流量，同时为每个租户部署一套租户 vRouter，转发租户内跨子网的流量。每个租户可以拥有多个租户子网，每个租户子网接入一个 vSwitch 并绑定一个 K8S 命名空间（NS）。租户子网 vSwitch 接入租户 vRouter，租户 vRouter 接入集群 vRouter，而集群 vRouter 接入全局 vRouter，从而实现租户内流量互通，而租户间的流量隔离。多租户集群组网如图 7 所示。

## 5 融合云原生基础设施验证及场景分析

基于 K8S 的融合云原生基础设施其管理面、控制面、工作面、存储面和网络面均可容器化部署在相同的物理机上，为了验证融合云原生基础设施的管理资源开销和响应速度，在实验室搭建了由 8 台物理机（24 核 CPU、128 GB 内存、4 TB SSD 存储）构成的融合云原生基础设施测试环境，合计 192 核 CPU、1 024 GB 内存和 32 TB SSD Ceph 存储。在云原生基础设施上创建了 3 个 KoK 集群、50 个 K8S 虚拟机、50 个 Kata 安全容器和 300 个裸机容器，使基础设施的整体资源负载达到 50%，基础设施管理面、控制面、工作面、存储面和网络面的系统资源开销见表 1。

可以看出，在上述实验室融合云原生基础设施中，所有平台基础组件的 CPU 资源开销为 8.2%，内存资源开销为 9.9%，基础设施组件本身的资源开销非常小，这也意味着即使是小型的云原生基础设施，采用融合云原生方案可分配的资源也在 90%以上，而对于裸机 K8S 和 KoO 方案，控制面和存储面通常部署在独立的物理机中，可分配资源比例较低。

进一步从资源支持类型、资源创建时间、性能损耗、弹性伸/缩和安全性等方面对比 4 种云原生基础设施方案见表 2。从支持的资源类型上看，融合云原生和 KoO 方案所支持的资源类型最多，

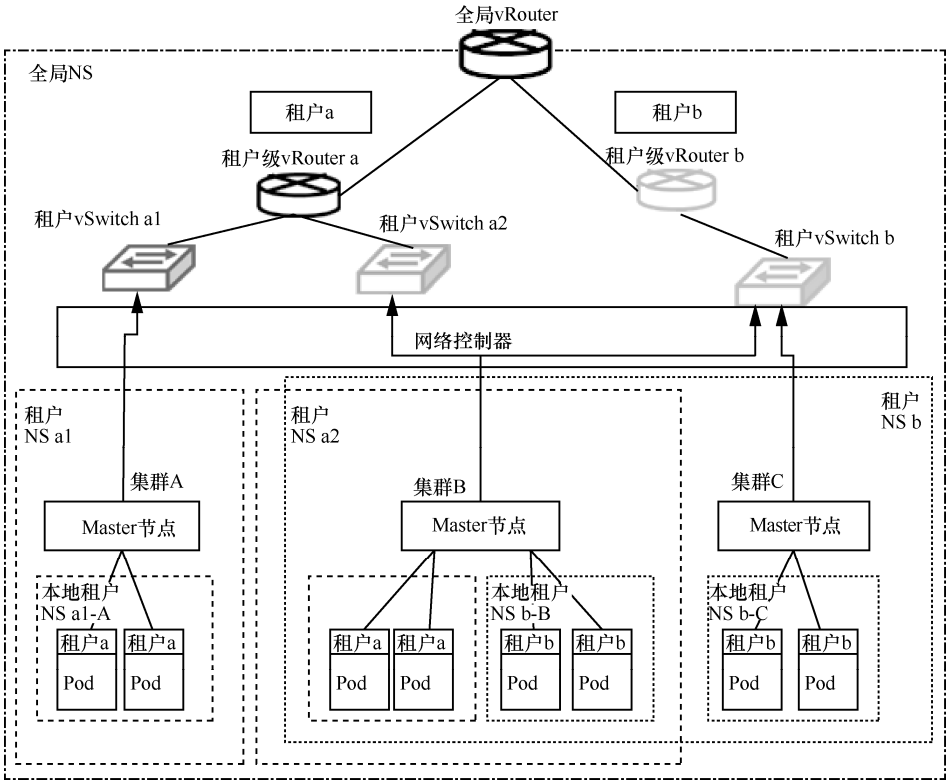



图 7 多租户集群组网

表 1　融合云原生基础设施管理资源开销

| 部件 | 单点 CPU/核 | 单点内存/GB | 总 CPU/核 | 总内存/GB | 节点数 |
|---|---|---|---|---|---|
| 管理面 | 0.4 | 1.2 | 0.8 | 2.4 | 2 |
| 控制面 | 2.2 | 1.1 | 6.6 | 3.3 | 3 |
| 工作面 | 0.35 | 0.1 | 2.8 | 0.8 | 8 |
| 存储面 | 0.6 | 11.6 | 4.8 | 92.8 | 8 |
| 网络面 | 0.1 | 0.2 | 0.8 | 1.6 | 8 |
| 资源合计 | | | 15.8 | 100.9 | |
| 资源占比 | | | 8.2% | 9.9% | |

表 2　几种云原生基础设施方案核心特性对比

| 项目 | 裸机 K8S | KoO | KoK | 融合云原生 |
|---|---|---|---|---|
| 支持的资源类型 | 裸机容器 | 裸机/虚拟机容器/虚拟机/安全容器/托管 K8S | 裸机容器/虚拟机/托管 K8S | 裸机/裸机容器/虚拟机容器/虚拟机/安全容器/托管 K8S |
| 虚拟机创建时间 | 不支持 | 分钟级 | 不支持 | 分钟级 |
| 容器创建时间 | 毫秒级 | 秒级 | 毫秒级 | 毫秒级 |
| 安全容器创建时间 | 不支持 | 秒级 | 不支持 | 亚秒级 |
| 托管 K8S 集群创建时间 | 不支持 | 分钟级 | 分钟级 | 分钟级 |
| 平均网络 I/O 损耗 | <10% | >60% | <20% | <20% |
| 容器/虚拟机直接路由 | 不支持 | 支持 | 不支持 | 支持 |
| 集群弹性伸缩 | 不支持 | 支持 | 支持 | 支持 |
| 安全性 | 容器+K8S 原生安全 | 虚拟化+多租户+K8S 原生安全 | 容器+K8S 原生安全 | 多层面多租户安全 |

而且融合云原生还支持更灵活的托管 K8S 集群模型；从资源创建时间上看，融合 K8S、裸机 K8S 和 KoK 的容器创建时间相当，均优于 KoO 方案；从网络性能损耗上看，裸机 K8S 性能最优，融合云原生与 KoK 性能相当；从安全性上看，融合云原生方案采用多层面多租户安全体系，强化了云原生基础设施的安全能力。

从上述分析可知，融合云原生方案与其他 3 种方案相比，其核心的优势在于资源开销少、资源选择广、安全系数高，其适合的应用场景包括：资源受限的云原生边缘云、边缘站点或研发测试云，能极大提升边缘云的可分配资源比例；多租户公有云或混合云的云原生基础设施，可提供租户隔离的云原生资源；多厂商、多专业混合承载的云原生网络云，通过融合了虚拟机、容器和安全容器的基础设施支撑各类云原生网元灵活承载。

## 6　结束语

基于 K8S 的融合云原生基础设施能够为网络云原生应用提供多样、高效、灵活的云原生基础资源，能够满足网络云极致的性能需求和边云协同的部署需求，有助于破解网络云多厂商统一承载难题，提升网络云原生业务的承载能力，推动网络云向云原生路线转型演进。

## 参考文献：

[1] CHUN B, HA J H, OH S, et al. Kubernetes enhancement for 5G NFV infrastructure[C]//Proceedings of 2019 International Conference on Information and Communication Technology Con-

vergence (ICTC). Piscataway: IEEE Press, 2019.
[2] Analysis and evaluation of Kubernetes based NFV management and orchestration[Z]. 2019.
[3] DAB B, FAJJARI I, ROHON M, et al. Cloud-native service function chaining for 5G based on network service mesh[C]//Proceedings of 2020 IEEE International Conference on Communications (ICC). Piscataway: IEEE Press, 2020.
[4] 陆钢, 陈长怡, 黄泽龙, 等. 面向云网融合的智能云原生架构和关键技术研究[J]. 电信科学, 2020, 36(9): 67-74.
LU G, CHEN C Y, HUANG Z L, et al. Research on intelligent cloud native architecture and key technologies for cloud and network integration[J]. Telecommunications Science, 2020, 36(9): 67-74.
[5] 苏坚. 基于云原生计算的 5G 网络演进策略[J]. 电信科学, 2018, 34(6): 147-152.
SU J. Evolution strategy of 5G network based on cloud-native computing[J]. Telecommunications Science, 2018, 34(6): 147-152.
[6] 李铭轩, 童俊杰, 刘秋妍, 等. 基于云原生的 5G 核心网演进解决方案研究[J]. 信息通信技术, 2020(2): 63-69.
LI M X, TONG J J, LIU Q Y, et al. Research on 5G core network evolution solution based on cloud native[J]. Information and Communications Technologies, 2020(2): 63-69.

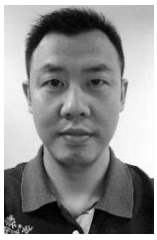
[作者简介]

**何震苇**（1976− ），男，中国电信股份有限公司研究院工程师，主要从事云原生技术、分布式架构、NFV、5G 等的研发工作。

**黄丹池**（1988− ），女，中国电信股份有限公司研究院工程师，主要从事云计算技术、容器技术方面的研发工作。

**严丽云**（1985− ），女，中国电信股份有限公司研究院工程师，主要从事云计算技术、容器技术方面的研发工作。

**林园致**（1996− ），女，中国电信股份有限公司研究院研究员，主要从事容器技术、DevOps 方面的工作。



**杨新章**（1972− ），男，博士，中国电信股份有限公司研究院高级工程师，主要从事业务平台、云计算技术、5G 方面的工作。