Kubernetes 容器云平台多租户方案研究与设计

黄丹池，何震苇，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：在容器云平台中，租户共享底层的计算、存储、网络等资源，存在租户容器运行和数据安全问题。分析了 Kubernetes 访问控制和资源隔离实现方案基础上，提出了一种基于多租户访问控制模型的容器云平台多租户方案，涵盖多租户管理模型、多租户访问控制、计算资源隔离和网络资源隔离等，可切实提升基于 Kubernetes 的容器云平台的资源隔离能力，有效降低数据安全隐患。

关键词：多租户；Kubernetes；容器
中图分类号：TN929.5
文献标识码：A
doi: 10.11959/j.issn.1000−0801.2020187

Research and design of multi-tenant scheme for Kubernetes container cloud platform

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

Abstract: In the container cloud platform, tenants share the underlying computing, storage, network and other resources, and there are problems with the operation of the tenant container and data security. Based on the analysis of Kubernetes scheme based on access control and resource isolation, a multi-tenant cloud platform was proposed, which covers multi-tenancy management model, multi-tenant access control, computing resources isolated and network resources, etc. The proposed model can be practically improving resource isolation capability based on containers of Kubernetes cloud platform, effectively reduce the data security hidden danger.

Key words: multi-tenant, Kubernetes, container

1　引言

云计算技术作为新一代的信息基础设施已成为广泛共识。随着云原生、微服务化等技术的不断发展，越来越多的企业开始采用基于 Kubernetes 技术构建容器云平台。目前，Kubernetes 在多租户场景的应用方面存在一些不足，例如没有一个统一的租户管理系统对租户权限进行管理，不能适应复杂的多租户关系；基于 Namespace 的计算资源和网络资源隔离相对简单，不能有效避免多个租户共享底层资源所引发的容器运行安全和数据安全问题。本文围绕 Kubernetes 分析其在访问控制和资源隔离的实现方案及其存在问题的基础上，提出一种基于多租户访问控制模型的容器云

收稿日期：2020−03−14；修回日期：2020−07−02

平台多租户方案。该方案涵盖多租户管理模型、多租户访问控制、计算资源隔离和网络资源隔离，切实提升基于 Kubernetes 的容器云平台的资源隔离能力，有效降低租户容器应用的数据安全隐患。

2　Kubernetes 多租户现状分析

容器云平台在多租户模式下每个租户都共用一套底层软硬件资源。为了限制租户之间对其他租户的资源和数据的访问，容器云平台需提供计算资源隔离、网络资源隔离和租户访问控制等功能，以确保租户在共享底层资源下的租户容器运行安全和租户数据安全。Kubernetes 集群会通过 Namespace 在用户间划分集群资源，对资源进行简单的逻辑隔离，这种隔离并不能满足容器云平台多租户的资源隔离需求。以下对 Kubernetes 访问控制和资源隔离作相应的分析。

2.1　Kubernetes 多租户访问控制分析

Kubernetes 的 API 访问请求都需要经过 Kubernetes API Server 提供的认证、授权和准入控制，只有全部通过才能获取到 API 请求的资源反馈。

（1）Kubernetes 认证

当 Kubernetes 集群开启 TLS 时，所有 API 请求都必须进行身份认证。Kubernetes 用户可以使用客户端证书、Bearer token、Bootstrap 引导令牌、Basic 静态密码文件、OpenID Connect ID token、ServiceAccount、webhook 等多种认证方式。认证成功后，用户信息包括用户名、UID、组名和额外字段等信息会传入授权模块做进一步授权验证。

（2）Kubernetes 授权

Kubernetes 通过策略进行授权请求，通过将请求的属性与访问策略进行比较，只有满足策略的请求才能被处理。若授权成功，该请求会发送到准入控制做进一步验证。

（3）Kubernetes 准入控制

准入控制是一个插件列表，在新建、修改和删除资源对象前会调用准入控制插件来验证操作的合法性，发送的请求必须通过列表中每个插件的验证，只有所有插件通过才可准入，若有一个插件验证失败则准入失败并返回错误。

Kubernetes 对 API 的访问执行了严格的控制，但对外部用户只执行了用户 API 请求的访问控制，通过从用户发起的 API 中获取用户凭证，如 Kube-config 中的证书或 Pod 中引入的 ServiceAccount，将用户凭证身份转化为 Kubernetes 中对应的 user 和 group。而对用户之间 API 的相互访问没有进行隔离控制，因此在外部用户 API 访问控制上还需要进一步隔离，限制其他租户用户对不是自身 API 的访问。

2.2　Kubernetes 多租户资源隔离分析

Kubernetes 主要通过 Namespace 来划分多租户的 Kubernetes 集群资源，结合多租户认证、授权、网络和资源配额等资源访问控制机制实现基本的资源隔离。

Kubernetes 控制平面有如下隔离功能。

- Namespace：Namespace 是 Kubernetes 的逻辑管理单位，用于将 Kubernetes 集群资源划分给不同租户以便不同租户在共享集群资源下统一管理。
- ResourceQuota：ResourceQuota 是 Namespace 级别的资源限制，对 Namespace 进行配额管理，确保 Namespace 下的资源使用不会超过其限制的值。
- LimitRange：LimitRange 是 Pod 和 container 级别的资源限制，即设置 Pod 中容器请求资源和不能超过 limit 的最大值和 request 的最小值。
- RBAC：RBAC 是 Kubernetes 授权模式之一，是基于角色的权限控制，允许管理员通过 KubernetesAPI 动态配置策略，集群资源对象通过 RBAC 来授权安全策略。
- ServiceAccount：相对于 UserAccount，

ServiceAccount 为 Pod 中的进程和外部用户提供身份认证信息，只有通过认证的 ServiceAccount，并经过授权和准入控制后才能访问 Kubernetes 集群中的资源。

- Secret：Secret 是 Kubernetes 中动态配置管理中的一种，解决密码、token、密钥等敏感数据配置问题。每个 Namespace 下都有一个 secret，当 Pod 启动时，该 secret 会自动 mount 到 Pod 指定目录中，Pod 容器进程使用该 secret 作为身份认证令牌访问 API server。

Kubernetes 数据平面有如下隔离功能。

- SecurityContext：安全上下文，用于定义 Pod 和 container 的权限和访问控制，设定安全相关配置，目的是限制不可信容器行为，保护系统和其他容器不受影响。
- NetworkPolicy：网络策略，支持按 Namespace 和 Pod 级别进行网络访问控制，使用 label 来指定 Namespace 和 Pod，只有匹配规则的流量才能进入或离开 Pod。

Kubernetes 在控制平面的隔离能力较强，而数据平面的隔离能力在要求数据安全性强的场景中是不足的，如金融、银行、大数据应用等多租户场景。随着电信运营商业务量增长，信息和数据逻辑集中程度越来越高，风险程度也不断提高，同时各分支机构系统进一步云化，通过容器技术在容器云平台中部署应用虽然能有效降低成本开支，实现资源的高利用率，但多租户问题也应运而生。目前，运营商正在实施网络重构战略，而容器云平台能有效承载各个系统部署和管理，但在云平台中如何有效地隔离租户资源成了待解决的另一问题，不仅要保障各个系统的容器应用运行安全，还要保障租户数据安全。为了保障各个系统有效稳定的运行在容器云平台中，容器云平台需要提供底层资源的硬隔离以保障租户容器运行的高安全隔离。本文将结合租户管理以及访问控制机制来隔离不同租户对资源的访问，以保障租户间共享底层资源下的容器运行和数据安全问题。

3 Kubernetes 容器云平台多租户方案

容器云平台多租户总体方案主要涉及租户访问控制和资源隔离。访问控制包括租户管理、认证、授权等，资源隔离包括计算资源隔离和网络资源隔离。本节将介绍 Kubernetes 容器云平台多租户总体方案设计。

在 Kubernetes 容器云平台多租户系统中，访问控制中租户管理是首要的，容器云平台租户模型层级关系划分为超级管理员、租户管理员、项目管理员和普通用户。根据租户模型通过用户管理系统对所有租户内用户进行认证和授权，租户认证结合 OpenLDAP 统一管理用户及用户的认证，并结合 Kubernetes 的 OpenID connect ID token 认证机制实现租户在集群内的认证，通过 Kubernetes 原生的 RBAC 授权策略实现租户授权，以此完成租户资源访问控制，限制租户对其他租户资源的访问。

租户资源隔离涵盖计算资源隔离和网络资源隔离。计算资源隔离需要保障容器运行的安全，目前 Kubernetes 默认使用 Docker 容器运行时创建容器，而通过 Docker 创建的容器并不能保证容器运行安全，要满足容器运行安全需保障多租户计算资源的隔离，通过采用其他的容器运行时创建容器，比如 Kata、Gvisor 容器，本文在 Kubernetes 集群中使用 Kata 安全容器代替 Docker 创建容器，以达到隔离计算资源的目的。网络资源隔离对多租户系统同样重要，在 Kubernetes 集群中其原生的网络是一个全通的扁平化的网络，没有实现多租户网络隔离，目前 Kubernetes 自身有 NetworkPolicy 机制可以提供 Namespace 和容器级别的隔离，在此基础上结合 Contiv-VPP 网络方案实现租户网络资源隔离。Kubernetes 容器云平台多租户总体架构如图 1 所示。

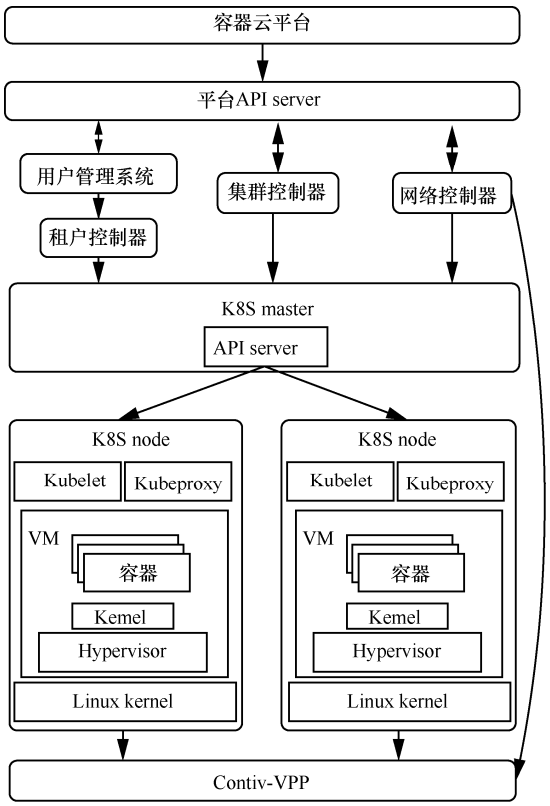


图 1　多租户总体架构

- 平台 API server：负责处理平台内 REST 操作，执行相关业务逻辑，是容器云平台的 API 入口。
- 租户控制器：负责租户管理、集群内租户认证、租户 RBAC、租户用户对资源的访问控制以及租户、项目资源配额的管理。在租户内用户分为：租户管理员、项目管理员和普通用户。集群租户认证是为外部用户访问集群时提供的认证，租户 RBAC 提供租户内用户与资源的权限绑定。通过租户控制器来完成租户用户认证授权和资源配额管理。
- 用户管理系统：外部第三方服务，提供访问容器云平台多租户系统的租户用户信息和认证服务。
- 集群控制器：负责租户在集群内的资源控制，通过在集群内创建集群租户角色，限制租户用户及租户内应用对 Kubernetes API server 的访问，用户访问集群资源、租户 Pod 访问 API server 需要通过身份令牌或 ServiceAccount 进行身份验证后才能访问。
- K8S API Server：Kubernetes 集群的 API 入口，租户集群控制面。提供整个租户集群资源对象增、删、改、查等接口，也提供其他模块之间的数据交互和通信，是资源配额控制的入口。
- K8S node：租户集群工作面。在工作节点上，通过隔离技术将租户计算资源和网络资源进行隔离。
- VM 虚拟机：由 Kata 安全容器创建 VM 虚拟机。Kata 创建的虚拟机为租户提供计算资源隔离，租户间容器应用所占用的计算资源互不影响。Kata 容器通过虚拟化技术（Hypervisor）启动一个轻量级虚拟机，虚拟机拥有单独的内核，使得每个容器都运行在一个具有单独内核的虚拟机内。
- Contiv-VPP：网络插件，提供网络资源隔离，通过网络控制器实现租户网络组网，为租户 Namespace 绑定租户网络，并通过 Kubernetes 的 NetworkPolicy 设置网络策略，为每个租户 Pod 资源提供网络端点间的通信和隔离。

4　Kubernetes 容器云平台多租户访问控制方案

容器云平台多租户访问控制涉及租户及租户下用户管理，包括用户认证、授权以及准入控制。在 Kubernetes 中没有提供多租户管理，但提供相关的扩展接口，为了管理外部租户，本节介绍基于 openLDAP 技术进行用户集中管理，并结合 Kubernetes 的 OpenID connect ID token 认证机制实现租户的认证，同时在 Kubernetes 的 RBAC 授权基础上增加租户授权控制模块，通过租户控制

器来实现租户角色与资源的绑定，准入控制上继承 Kubernetes 原生的准入控制。

4.1 容器云多租户管理模型设计

多租户是指一个建立在共同的底层资源上的环境被多个租户共同使用，租户可以是一个组织、一个企业或企业下的某个部门或项目组或一个用户等。每个租户可以管理自己的集群资源，租户间通过 Namespace 进行隔离，每个租户拥有一个或多个租户级 Namespace，租户内有项目组，项目组之间也相互隔离，每个项目组划分为一个 project，一个 project 对应一个或多个租户级 Namespace，项目内成员拥有对项目资源的访问权限，通过租户控制器创建租户 RBAC，并通过租户 RBAC 对租户用户进行权限控制，以此隔离不同租户不同用户间的访问。多租户模型如图 2 所示。

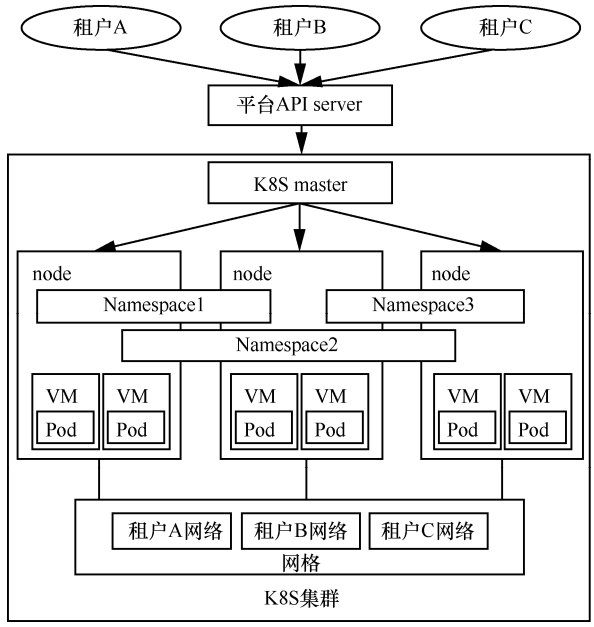


图 2 多租户模型

多租户共享集群，在控制面租户共享集群 master，在工作面租户通过 Namespace 进行资源隔离。租户由超级管理员创建并通过租户控制器管理和分配租户资源及配额，在租户内租户管理员创建项目组，为每个项目添加成员，成员用户为租户内用户。租户内有多个租户级 Namespace，项目组之间相互隔离并且每个项目组对应其中一个或多个租户级 Namespace，Namespace 下资源归项目组成员所有。租户内容器应用部署在通过 Kata 安全容器创建的虚拟机中，通过网络控制器为每个租户创建一个虚拟网络。结合 openLDAP 对租户、用户进行管理，每个租户、用户访问平台时都要进行认证和授权，通过角色绑定授予租户对资源的访问，设置资源访问策略结合资源隔离技术实现多租户资源隔离的目的。

4.2 租户管理和认证设计

在 Kubernetes 中 OpenID connect ID token 认证是通过第三方 Id provider（IdP）提供用户信息，通过外部用户管理系统对外部用户进行管理、认证和部分授权操作。Kubernetes 的 user account 是发生在 Kubernetes 系统之外，用户管理系统通过 openLDAP 进行管理，该系统内的用户都可以对 Kubernetes 资源进行访问，但能访问哪些资源需要授权后才能发起访问请求。OpenID connect ID token 是在 OAuth2 协议上的扩展，能在认证成功后获取 access token、ID token 和 refresh token，ID token 是一种 JSON Web token（JWT），通过解码后可以查看到 Token 内包含的用户信息，并在 Kubernetes 中使用该 ID token 访问 Kubernetes API server，同时使用该 token 进行后续授权操作。身份认证流程如图 3 所示。

4.3 租户授权控制设计

Kubernetes 本身只提供对 ServiceAccount 资源对象的管理，对于 UserAccount 外部账户并不在其管理范围内，而外部账户访问 Kubernetes 资源时需要对其进行认证、授权和在准入控制策略允许的情况下才能访问租户内资源。本节描述如何对多租户进行授权控制。

对租户进行层级划分，将用户分为超级管理员、租户管理员、项目管理员和普通用户。

（1）超级管理员：有且只有一个超级管理员用户，该用户拥有最高权限，拥有对整个平台和

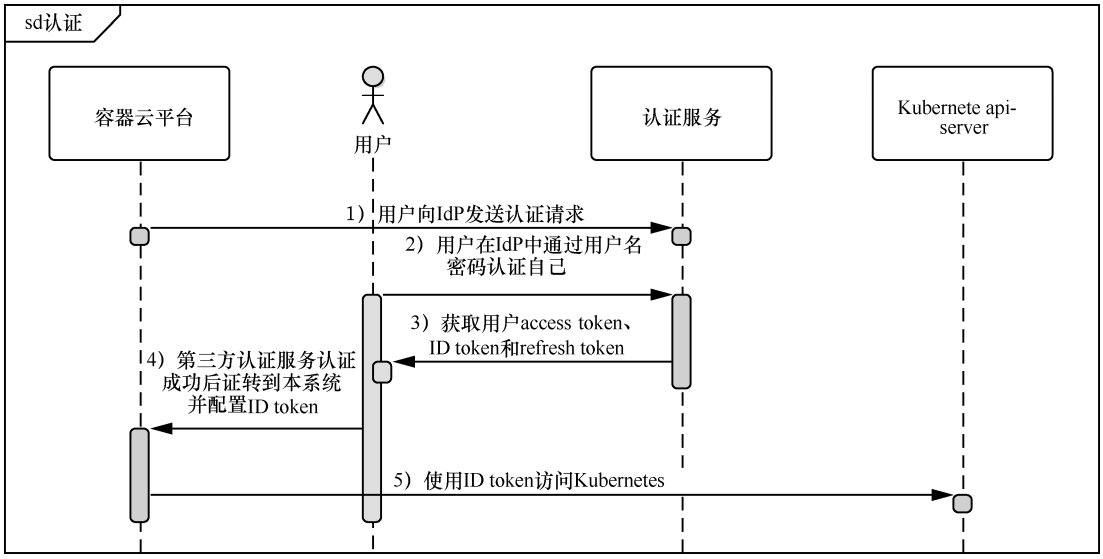


图 3　身份认证流程

集群资源的管理和控制，能创建租户用户、分配权限等。

（2）租户管理员：首先由超级管理员创建租户，然后指定该租户为租户管理员用户，该用户拥有租户下所有资源的权限，可以管理租户内项目、角色、权限的增加、读取、更新和删除操作即 CRUD，并赋予租户内用户权限等。

（3）项目管理员：属于项目级别的用户，在租户内用户创建项目，创建项目的用户为项目管理员，租户下有多个项目管理员，由租户管理员给项目分配资源配额，并和项目管理员负责项目成员添加，成员用户通过与项目 role 绑定获取对项目资源的访问权限。

（4）普通用户：通过与角色绑定，获取相应的权限。

多租户层级之间的关系如图 4 所示。

admin 属于最高权限，管理租户及租户下的用户，给租户分配租户资源配额，将租户与租户级 role 绑定实现租户资源控制；租户管理员拥有租户内最高权限且只属于一个租户，管理租户及租户下资源，包括用户、项目、角色、权限的 CRUD，并能将集群级 role 或 Namespace 级 role 通过 RoleBinding 将其绑定给某个用户。在一个企业或组织内，为了与其他团队或部门进行资源隔离，在租户内还需进一步隔离，在容器云平台中通过创建项目的方式进行租户内隔离和

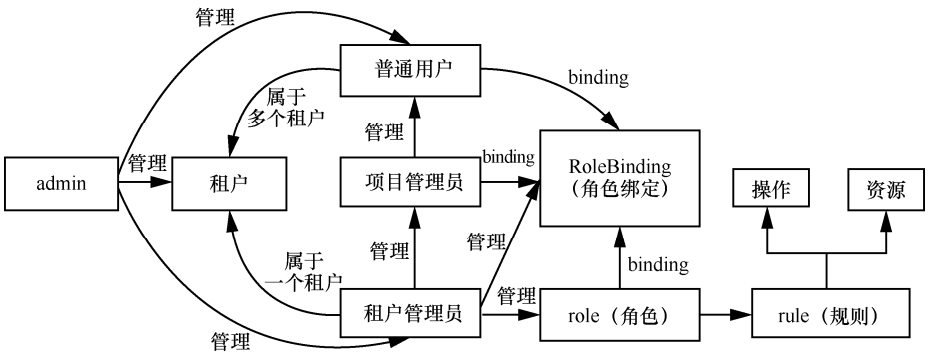


图 4　多租户层级关系

管理，由租户内的用户创建项目，并指定该用户为项目管理员，同时由租户管理员邀请普通用户到项目下，成为项目内成员；项目管理员拥有项目内最高权限，管理项目及其成员、角色的 CRUD 等；普通用户只能通过角色绑定来获取对资源的 CRUD 操作，但普通用户可以属于多个租户。

5 Kubernetes 容器云平台多租户资源隔离方案

在容器云平台中，资源隔离对多租户容器应用运行安全和数据安全至关重要，资源隔离中主要涉及计算资源和网络资源的隔离。通过第 2.2 节的分析，Kubernetes 集群在数据面的隔离能力比较弱，为了加强租户间的隔离，提高租户容器运行安全和数据安全，本节将介绍使用 Kata 容器来隔离计算资源和使用 Contiv-VPP 网络组网方案实现网络隔离。

5.1 计算资源隔离方案设计

多租户计算资源隔离需要做到容器运行时的安全隔离，本节介绍使用 Kata 容器来隔离容器运行时计算资源的隔离。Kata 容器是通过虚拟化技术解决容器内部安全问题的，底层采用 Hypervisor 技术，将容器作为轻量级虚拟机启动，启动虚拟机后再在虚拟机中运行容器，虚拟机相当于 Kubernetes 中的 Pod。

Docker 与 Kata 对比如图 5 所示，与 Docker 相比，Kata-container 和原来的 runc 是平级的，通过 Docker 启动容器可以替换为用 Kata 启动，Kata 容器解决了传统容器共享内核的安全和隔离问题，它主要是让每个容器运行在一个轻量级的虚拟机中，并使用单独的内核。Kata 可以实现和现在容器生态的无缝对接，与 Kubernetes 容器编排工具能很好的结合，提供容器的快速启动和虚拟机级别的安全隔离，Kata 与 Kubernetes 集成方式如图 6 所示。

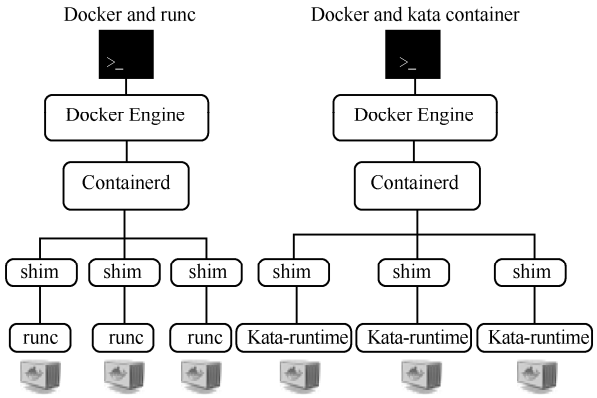


图 5 Docker 与 Kata 对比

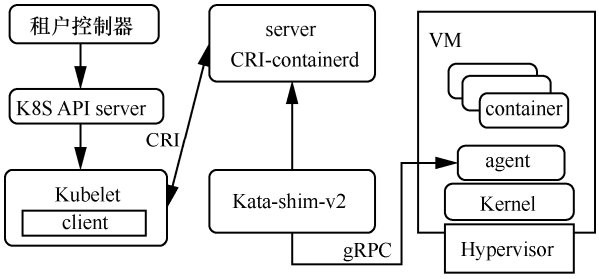


图 6 Kata 与 Kubernetes 集成

Kata-container 有 4 个组件分别为 shim、runtime、proxy 和 agent，agent 运行在虚拟机里，shim、runtime 和 proxy 运行在宿主机上，每次创建一个容器就会创建一个 proxy、agent 和 shim，而 runtime 只有一个。Kata 不与 Kubernetes 直接通信，只跟实现了 CRI 的容器管理进程打交道，Kubernetes 不提供容器运行时环境，但提供 CRI。agent 运行在虚拟机里，shim、runtime 和 proxy 运行在宿主机上，每次创建一个容器就会创建一个 proxy、agent 和 shim，而 runtime 只有一个。

Kubernetes 传统上是通过 Docker 来创建容器，并由容器提供 Pods 服务，但在多租户环境下，Docker 创建的容器可能会发生互相攻击，若攻击到内核，则所有容器都会崩溃，无法保证租户资源安全隔离和容器运行安全，因此用 Kata 安全容器代替 Docker 创建容器服务。在容器云平台中创建容器，集群控制器会向 Kubernetes 集群发送创建容器请求，在 Kubernetes API server 检测到创建

容器请求时，Kubelet 组件通过 CRI 向 CRI-containerd 请求执行创建容器操作，此时在 Kata 内通过 Kata-shim-v2 连接到 Kata-proxy 提供的 gRPC 服务上，再基于虚拟化技术（Hypervisor）启动容器运行所需的虚拟机，虚拟机包含 guest Kernel 和 guest image，guest Kernel 用于启动虚拟机，guest image 包含基于 initrd 和 rootfs 的最小映像，虚拟机启动后 Kata-proxy 会连接到虚拟机与 Kata-agent 通信，并配置虚拟机内部沙盒，然后会依据 OCI 配置 config.json 文件，并和 Kata-agent 通信创建容器，创建的容器运行在一个使用单独内核的超轻量级虚拟机中，完成容器的创建。

通过上述介绍，在容器云平台多租户系统中通过使用 Kata 安全容器代替 Dockerrunc 创建容器服务，能有效解决传统容器在共享宿主机内核等硬件资源上的安全和隔离问题，保证各个租户的容器应用互不影响，同时隔离租户间底层计算资源，切实保证租户容器运行安全和计算资源隔离。

5.2　网络资源隔离方案设计

Kubernetes 自身使用 flannel 插件实现网络模型，使得集群内 Pod 之间互通，无论是在节点内还是在节点外都可以互相通信。而在复杂的多租户环境下，租户应该有自己独立的网络，不同租户间网络隔离。Contiv-VPP 支持 NetworkPolicy 网络策略控制，并且是基于 VPP 和 DPDK 数据平面的开源容器网络方案，利用 VPP 网络栈全面接管基于 Linux 内核的 Kubernetes 原生网络，它是 Kubernetes 的一个网络插件，执行 Kubernetes 的 policy 和 service，支持 L2 和 L3 网络隔离，而 Kubernetes 自身的 NetworkPolicy 机制可提供 Namespace 和容器级别隔离，满足租户网络隔离需求。多租户网络资源隔离使用 Contiv-VPP 网络插件承载 Kubernetes 的网络环境，并通过租户网络组网方式实现租户间网络资源隔离。Contiv-VPP 在配置中会接管物理网卡，采用数据分组转发引擎 VPP 实现网络加速，提升网络性能，在进行网络资源隔离时保证租户在创建 Pod 时加入租户自身虚拟网络中，同时保障租户 Pod 流量快速转发。多租户网络拓扑关系如图 7 所示。

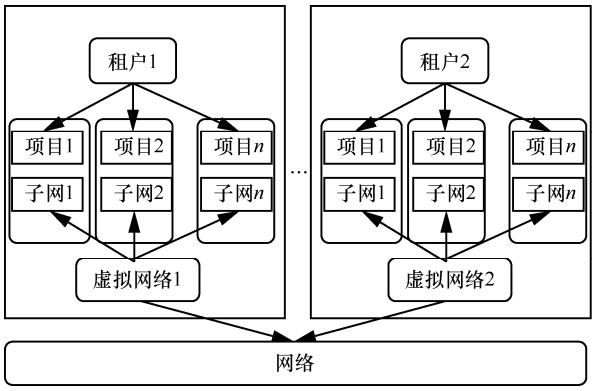


图 7　多租户网络关系

在创建租户时，为租户分配网络资源，给予不同租户不同网段网络，通过网络控制器给租户创建一个虚拟网络，实现租户间网络隔离。租户内的 Pod 资源运行在各自的租户虚拟网络下，在租户内创建项目时，给项目指定一个子网，每个项目分配不同的子网，实现租户内各项目间网络隔离，各租户间、租户内项目间通信需配置路由进行通信，否则租户间、项目间是相互隔离的。通过结合 Contiv-VPP 网络插件和 Kubernetes 的 NetworkPolicy 策略实现租户间网络隔离。租户网络组网架构如图 8 所示。

在容器云平台中有一个全局路由，平台会通过网络控制器为租户创建一个租户虚拟网络，同时创建租户路由，租户所有 Namespace 对应一个租户 Namespace 级的 vSwitch，挂在租户路由下，该 vSwitch 为租户 Namespace 建立独立的子网，以隔离租户不同 Namespace 的网络资源，租户内不同 Namespace 通信需通过租户路由和 vSwitch 连通租户内不同 Namespace。租户内有项目组，每个项目组对应一个或多个租户 Namespace，项目组内通信和隔离与租户 Namespace 通信和隔离类似，通过在租户路由上预先配置好策略，将项目内一个 Namespace 子网路由到另一个

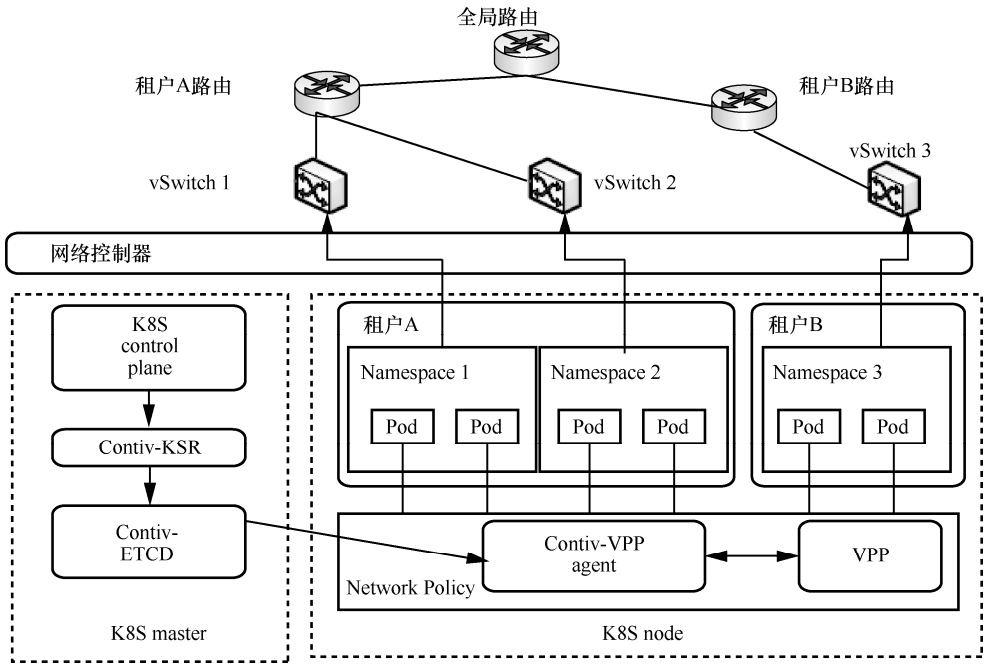


图 8　租户网络组网架构

Namespace 子网，实现跨 Namespace 通信。租户内同 Namespace 的 Pod 之间能互相通信，但可通过 NetworkPolicy 网络策略控制 Pod 间数据流量。数据会从 Contiv-VPP 策略插件中自顶向下进行转换，顶层接收来自 ETCD 的 Kubernetes 状态数据，并由 Contiv-KSR 从 Kubernetes API 中反映出来存储于 Contiv-ETCD 中，经过 Contiv-VPP/agent 将策略相关的数据转换为可使用的格式策略并输出到 VPP 中。在租户内创建 Pod 时指定网络资源以及 NetworkPolicy 策略，能实现 Pod 级的网络隔离，而租户内不同 Namespace 间子网隔离，通过在租户路由中的策略实现租户内不同 Namespace 通信。租户间通信通过全局路由实现网络互通。通过该组网方式能实现租户间网络资源隔离，保证租户访问网络资源的安全性，同时结合 Contiv-VPP 网络插件能保障高效的网络传输性能，实现网络加速。

6　结束语

在容器云平台多租户系统中，不同租户之间的资源隔离仅仅基于 Kubernetes 的 Namespace 只能进行简单的弱隔离，且 Kubernetes 自身并不支持多租户的管理。本文在 Kubernetes 基础上研究和设计了多租户的访问控制和资源隔离方案，通过使用 Kata 安全容器和 Contiv-VPP 网络组网方案实现多租户的访问控制和租户资源隔离，能有效提高容器云平台多租户平台资源隔离能力，切实保障租户容器应用运行安全和租户资源隔离。在电信运营商实施的网络重构战略中，多租户技术对运营商网络重构、系统整合中具有重要的战略价值，通过容器云平台部署和管理运营商各分支机构下的子 IT 系统，使容器应用运行在资源隔离的容器云平台中，不仅能在保障容器运行安全下对各系统集中管理，还能降低各系统资源部署开销，提升资源利用率。

参考文献：

[1] 李伟东，杨小虎. 基于 Kubernetes 的容器云平台强多租户模型关键技术研究[D]. 杭州: 浙江大学, 2019: 1-63.
LI W D, YANG X H. Key techniques of hard multi tenancy model in Kubernetes based container cloud[D]. Hangzhou: Zhejiang University, 2019: 1-63.

[2] 袁雪波, 丁旭阳. 基于 OpenStack 的多租户数据安全保护技术研究[D]. 成都: 电子科技大学, 2017: 1-58.
YUAN X B, DING X Y. Research on multi-tenant data security protection technology based on OpenStack[D]. Chengdu: School of Computer Science and Engineering, 2017: 1-58.
[3] BobKillen, Liggitt. Authenticating[EB]. 2020.
[4] Egernst, Nitkon, Jodh-intel, GabyCT. Kata containers architecture[EB]. 2019.
[5] Rastislavszabo. Contiv/VPP Kubernetes network Plugin[EB]. 2019.
[6] Rastislavszabo, Milanlenco, Jmedved. Kubernetes network policies in Contiv/VPP[EB]. 2019.
[7] 朱瑜坚, 马俊明. 一种面向多租户的 Linux 容器集群组网方法[J]. 计算机科学, 2018(9): 46-51.
ZHU Y J, MA J M. Linux container cluster networking approach for multiple-tenants[J]. Computer Science, 2018(9): 46-51.
[8] 杨约社, 戴元顺. 云环境下多租户的网络隔离的研究与实现[D]. 成都: 电子科技大学, 2019: 1-53.
YANG Y S, DAI Y S. Research and implementation of multi-tenant network isolation in cloud environment[D]. Chengdu: School of Computer Science and Engineering, 2019: 1-53.

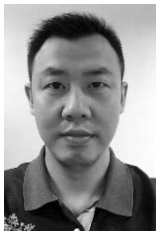
[作者简介]

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。

何震苇（1975– ），男，中国电信股份有限公司研究院工程师，主要从事容器、PaaS、分布式架构、NFV 等的研发工作。

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、CICD 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、移动互联网方面的工作。