CNUTCon 2017 全球运维技术大会

多租户Kubernetes实践
——从容器运行时到SDN

倪朋飞

HyperHQ & Kubernetes Maintainer

主办方 Geekbang 极客邦科技 InfoQ

Geekbang.
极客邦科技

极客搜索

全站干货，一键触达，只为技术

s.geekbang.org

扫描二维码立即体验

有没有一种搜索方式，能整合 InfoQ 中文站、极客邦科技旗下12大微信公众号矩阵的全部资源?
极客搜索，这款针对极客邦科技全站内容资源的轻量级搜索引擎，做到了!

扫描上方二维码，极客搜索!

EGO EXTRA GEEKS' ORGANIZATION NETWORKS

这里只有
技术领导者

EGO会员第二季招募季正式开启

报名时间：9月1日-9月15日

扫描添加E小欧，

邀您进入EGO会员预报名群

立即报名

E小欧

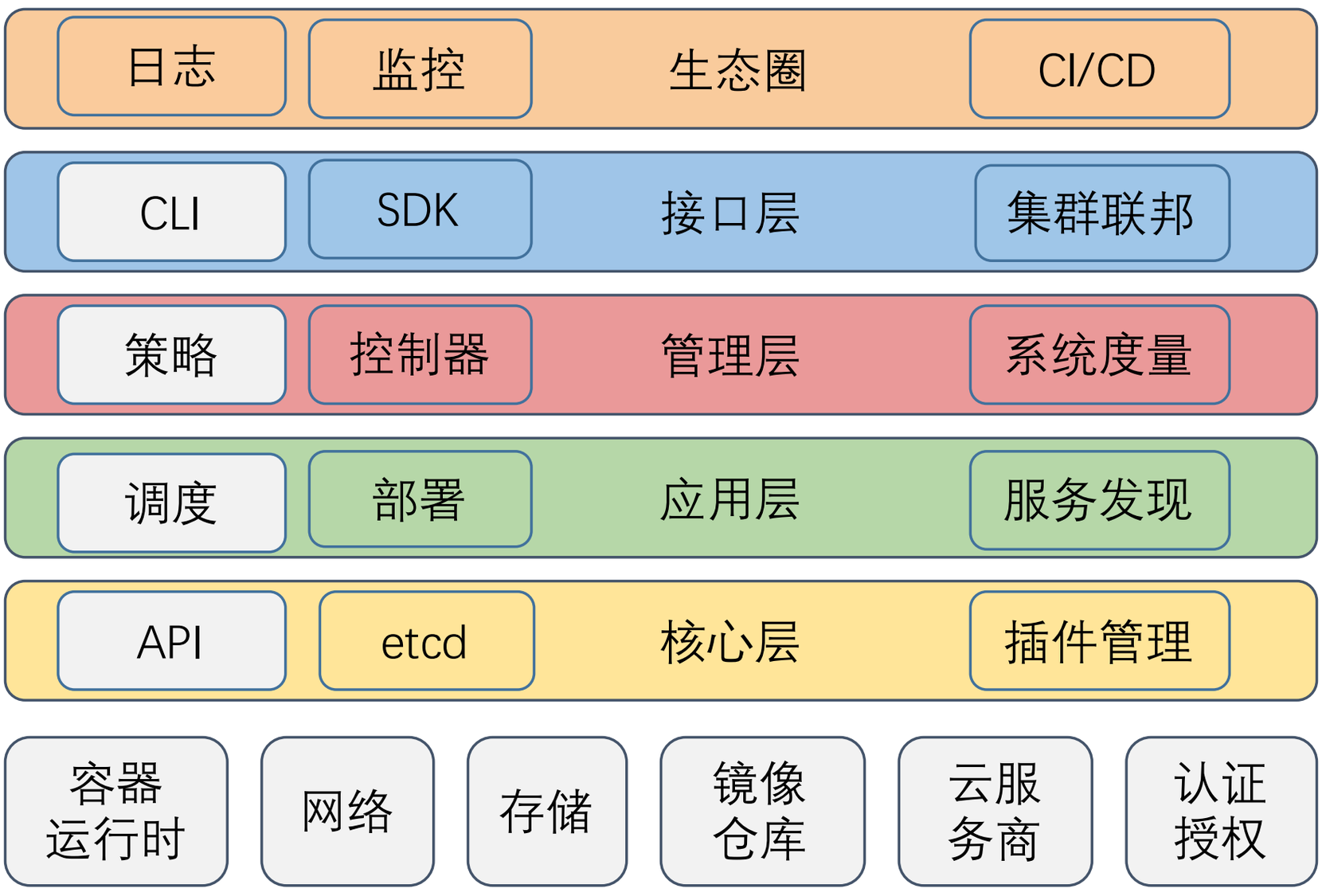
TABLE OF **CONTENTS**

TABLE OF **CONTENTS**

Kubernetes简介

- Google开源的容器集群管理系统，CNCF核心项目
- 最流行、最活跃的容器编排系统
- 完善的容器集群管理能力
  - 访问控制、服务发现、自动故障修复、滚动升级等
- 简单易扩展的架构设计及丰富的插件支持

Kubernetes插件机制



Kubernetes插件示例

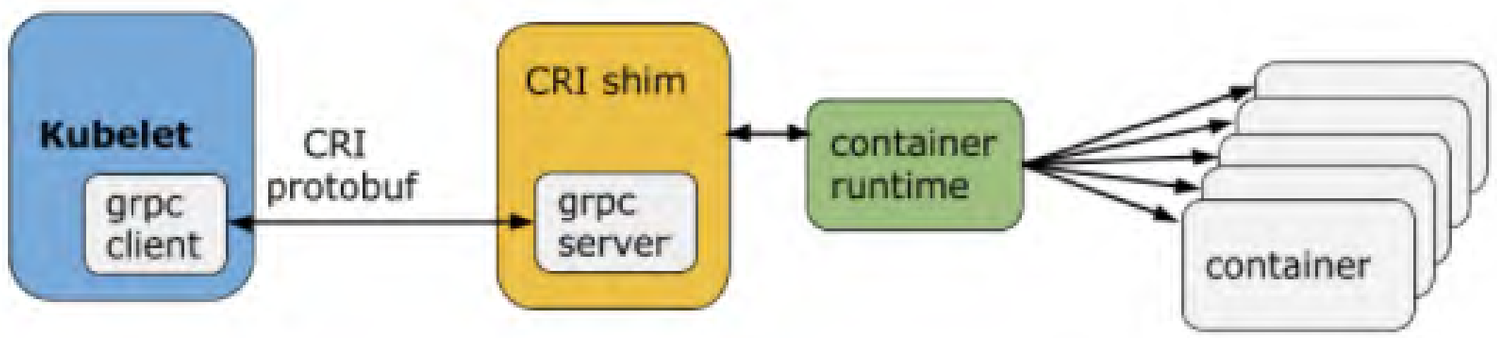
- 认证：X509、Token、密码等
- 授权：ABAC、RBAC、OpenID、Webhook等授权
- 准入控制：ServiceAccount、LimitRanger、ResourceQuota等
- 调度：查询未调度的Pod并更新Node绑定
- 存储卷：hostPath、NFS、rbd等内置插件以及FlexVolume扩展
- 网络：kubenet、Container Network Interface (CNI)
- 容器运行时：Container Runtime Interface (CRI)
- 云提供商（Cloud Provider）

TABLE OF **CONTENTS**

Container Runtime Interface

- Container Runtime Interface (CRI) 是一个基于gRPC的容器运行时接口
- 外部容器运行时实现gPRC服务端，kubelet作为客户端访问
- 无需关心内部通信逻辑，只需实现RuntimeService和ImageService接口
- 容器运行时负责管理容器网络，如使用CNI等
- 社区多种实现，如frakti、cri-containerd、cri-o等
- kubelet启动时配置

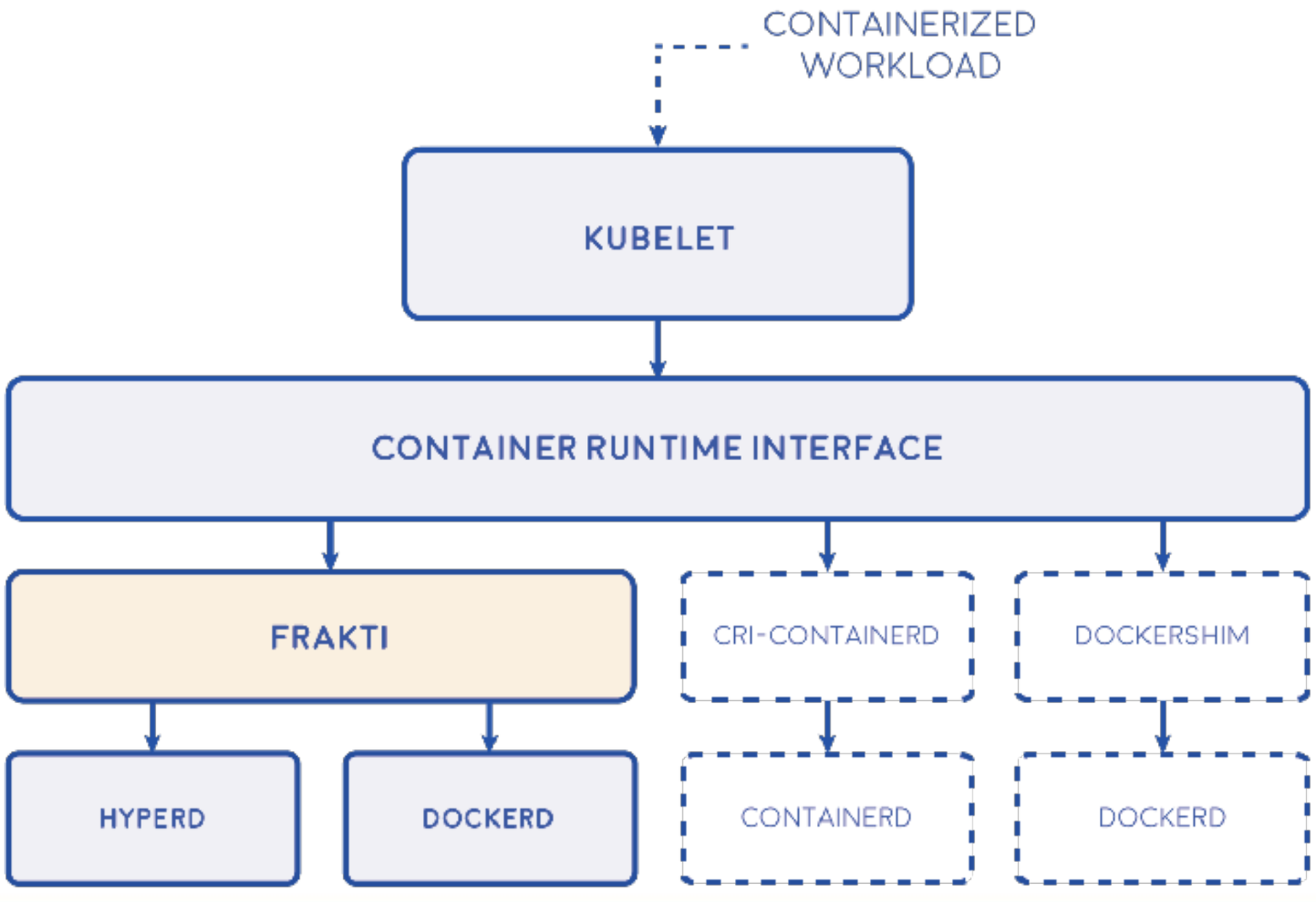
  container-runtime-endpoint.



Frakti

- 基于Kubelet CRI的虚拟化容器运行时
- Pod运行在单独的虚拟机中，属于同一个Pod的容器运行在同一个虚拟机中
- 基于CNI的容器网络
- 原生支持Kubernetes社区的各种扩展和工具
- 目前状态：Kubernetes v1.7 beta版和v1.6 alpha版
- 开源：https://github.com/kubernetes/frakti

Frakti


CONTAINERIZED
WORKLOAD
KUBELET
CONTAINER RUNTIME INTERFACE
FRAKTI
HYPERD
DOCKERD
CRI-CONTAINERD
CONTAINERD
DOCKERSHIM
DOCKERD

HyperContainer

- 基于虚拟化的容器引擎，支持KVM、XEN、Libvirt等
- 亚秒级启动（<100ms Xeon 1270，~400ms Pine64 ARM）
- 多平台支持，如x86、ARM、Power等
- 独立内核，无需Guest OS
- 简单易用
- 开源：https://github.com/hyperhq/hyperd

HyperContainer vs Docker

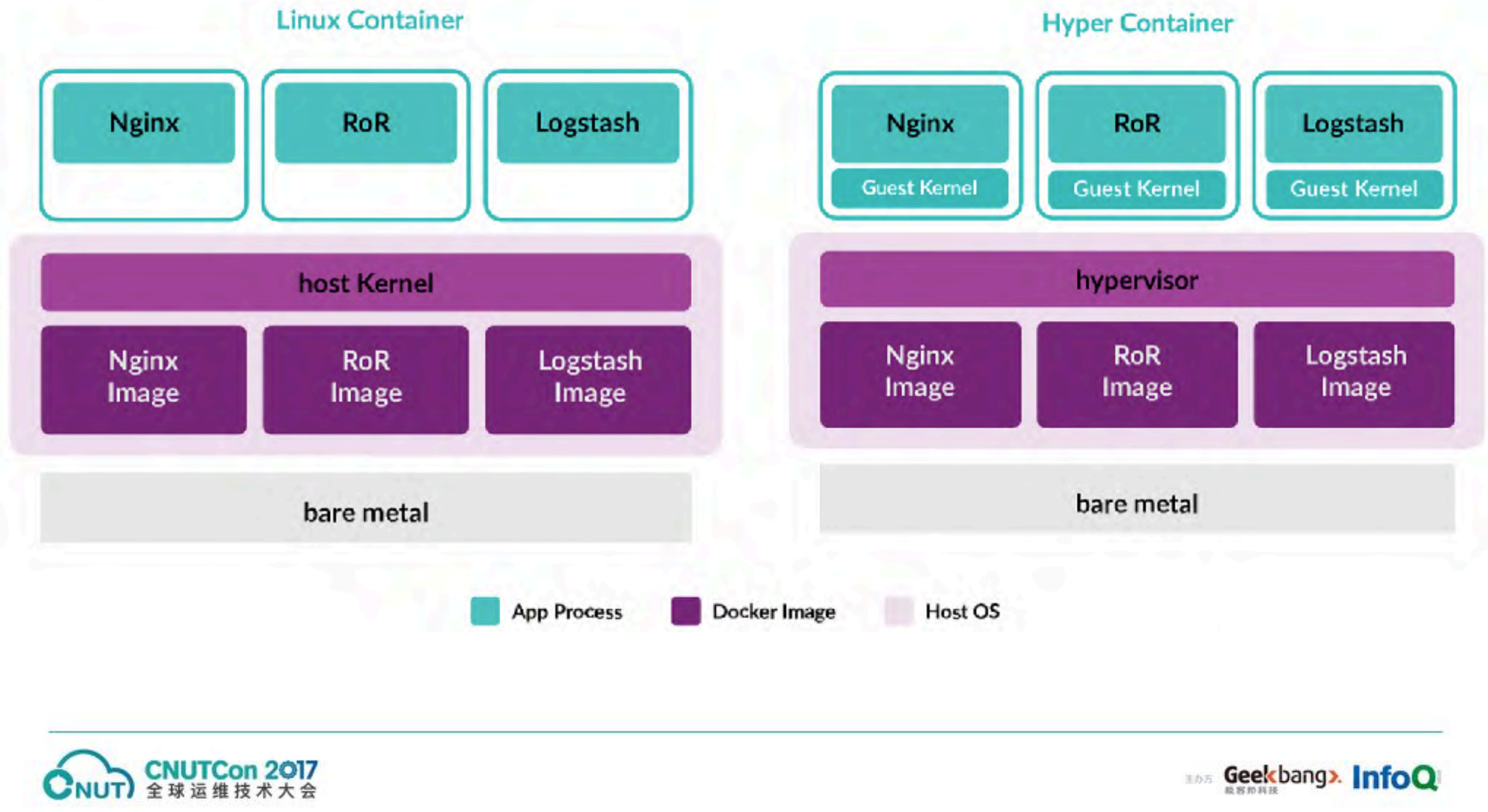


TABLE OF **CONTENTS**

Kubernetes网络模型

- Kubernetes网络模型
  - IP-per-Pod，每个Pod都拥有一个独立IP地址，Pod内所有容器共享一个网络命名空间
  - 集群内所有Pod都在一个直接连通的扁平网络中，可通过IP直接访问
  - Service clusterIP在集群内部访问，外部请求需要通过NodePort、LoadBalance或者Ingress来访问
- 网络插件
  - Container Network Interface (CNI)：容器网络接口，CNCF核心项目
  - kubenet：基于CNI bridge的网络插件，扩展了端口映射、QoS等功能

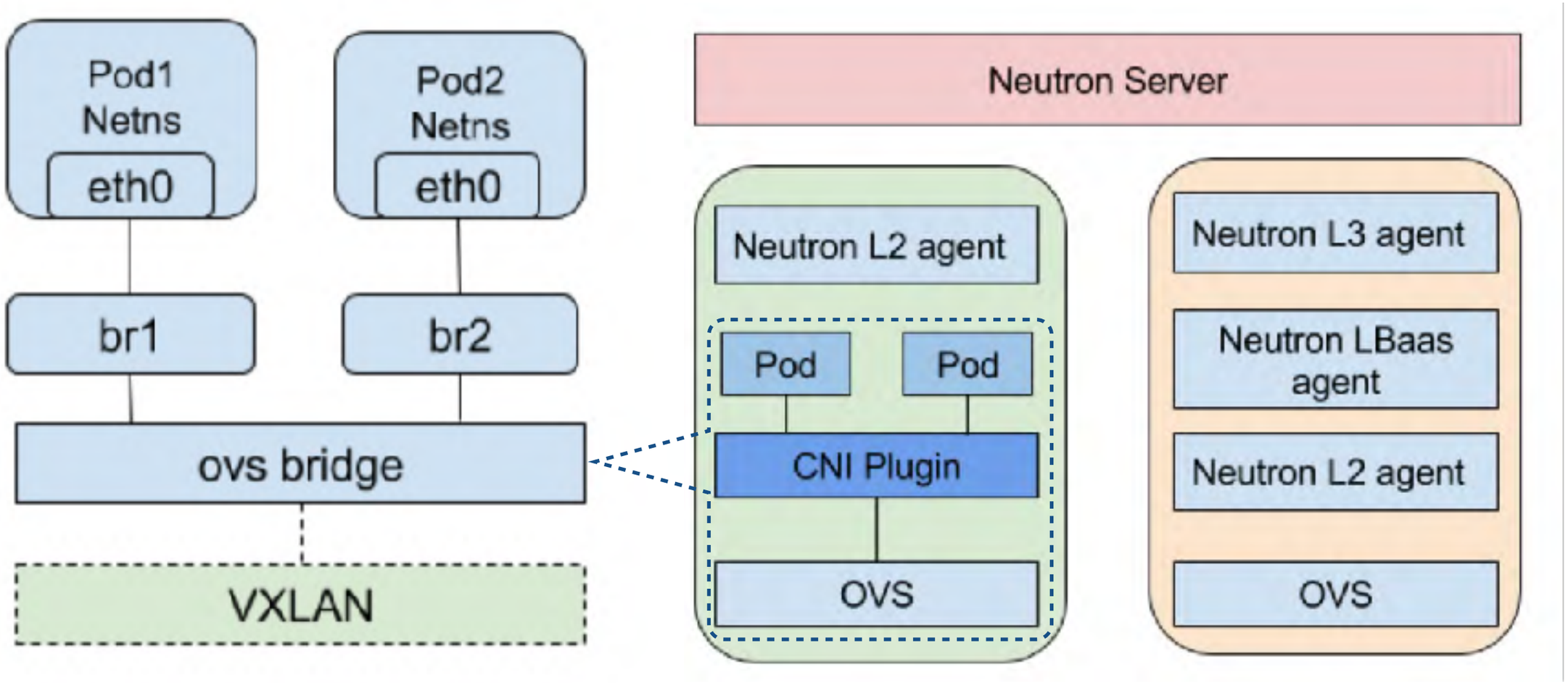
Container Network Interface

- Kubernetes网络插件的基础，CNCF核心项目
- 基本思想：容器运行时在创建容器时，先创建network namespace，然后调用CNI插件为其配置网络，最后再启动容器内的进程
- CNI插件组成
  - CNI Plugin负责给容器配置网络（如bridge、macvlan、calico等）
  - IPAM Plugin负责给容器分配IP地址（如host-local、dhcp等）

基于CNI的容器SDN网络

- 多网络管理
  - CustomResourceDefinition (CRD)管理网络对象
  - 自定义控制器监听网络变化并在Neutron中操作
  - Annotation将网络名称传给CNI网络插件
- CNI网络插件
  - 向Neutron查询网络参数
  - 配置Pod连接到Neutron openvswitch插件
- 开源：https://git.openstack.org/cgit/openstack/stackube

基于CNI的容器SDN网络



Network示例

```
[node1 ~]$ cat network-resource.yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: networks.stackube.kubernetes.io
spec:
  group: stackube.kubernetes.io
  names:
    kind: Network
    listKind: NetworkList
    plural: networks
    singular: network
  scope: Namespaced
  version: v1
[node1 ~]$ kubectl create -f network-resource.yaml
customresourcedefinition "networks.stackube.kubernetes.io" created
[node1 ~]$ cat network.yaml
apiVersion: stackube.kubernetes.io/v1
kind: Network
metadata:
  name: demo
spec:
  cidr: 10.244.0.0/16
  gateway: 10.244.0.1
[node1 ~]$ kubectl create -f network.yaml
network "demo" created
[node1 ~]$ kubectl get network
NAME      KIND
demo      Network.v1.stackube.kubernetes.io
```

TABLE OF **CONTENTS**

实践经验1

问题1: 如何支持多租户的服务发现和负载均衡?

- 挑战
    - 租户只可访问自己的服务
    - 相同网络上的Pod才可以相互访问
- 解决方法
    - 自定义kube-proxy，仅监听租户的服务
    - 基于neutron lbaas提供服务的负载均衡

实践经验2

问题2: 如何支持多租户网络的DNS解析?

- 挑战
    - 租户只需要访问自己的service
    - 租户网络与HostNetwork不通
- 解决方法
    - 自定义kube-dns，仅监听租户的services和endpoints
    - 以普通Pod方式将kube-dns运行在租户网络内

实践经验3

问题3: 如何管理Kubernetes系统服务（如kube-proxy、apiserver等）？

- 挑战
  - 加入hypervisor后，无法支持hostnetwork以及特权容器
  - kubernetes系统服务需要访问并修改hostnetwork
- 解决方法
  - 混合容器运行时（如HyperContainer+Docker），将租户的普通容器运行在虚拟机中而系统服务运行在Docker容器中
  - 所有容器运行时共享相同的CNI网络

THANKS!

智 能 时 代 的 新 运 维

CNUTCon 2017