NJSD 全球软件大会 2017
IT大咖说 知识分享平台

# 容器网络助力原生云

曹水
华为 中央软件院
HUAWEI

3rd NJSD
Global Software Development Conference . Nanjing
全球软件大会
2017

# What’s Container Network

Container Network provides communication about container-to-container and container-to-external network.

A Container Network needs to solve the following:

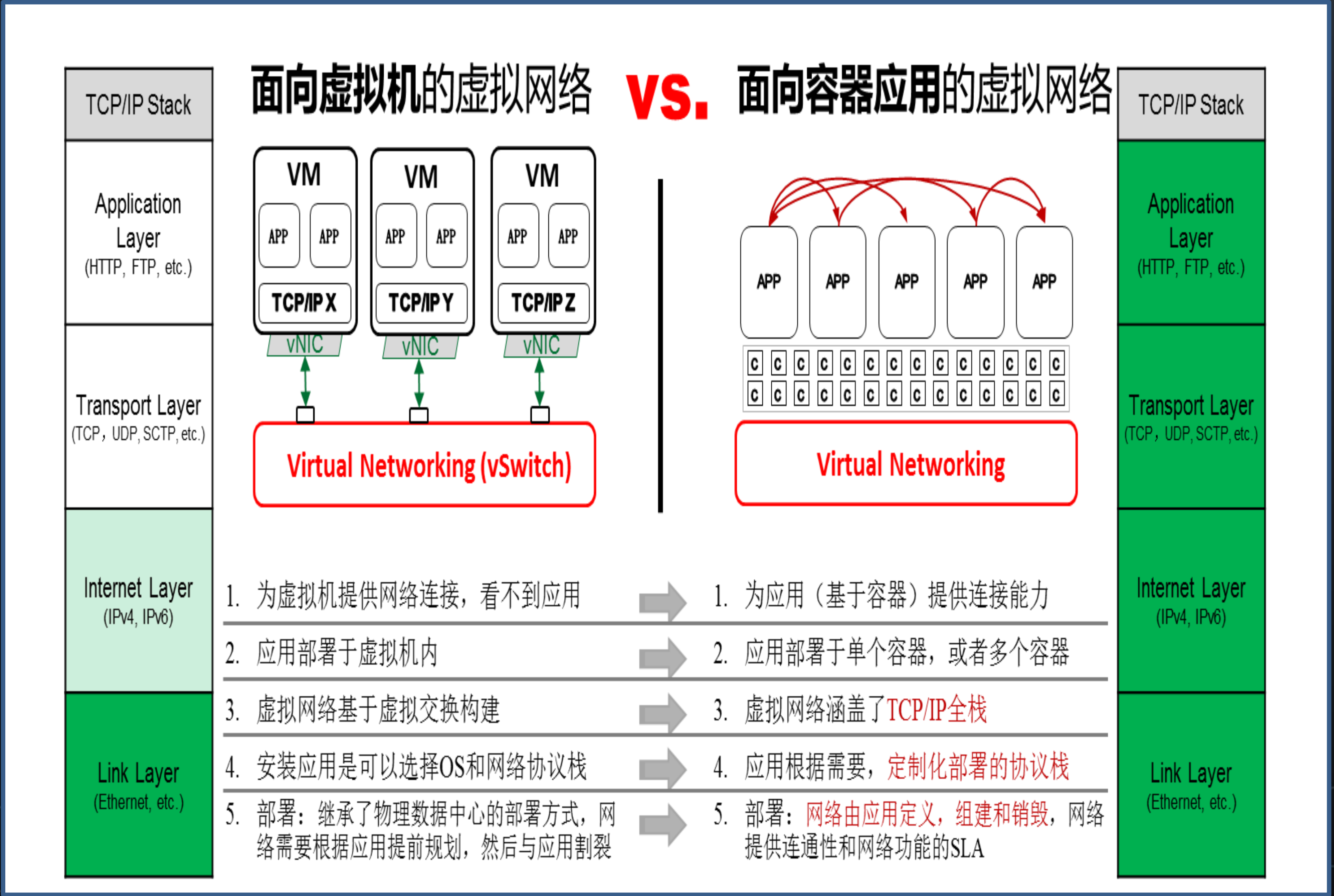
✓ Container Network Specifications

✓ IP/MAC address allocation

✓ Router Rules

✓ Data Plane selection

# The Nature of Container Network



面向虚拟机的虚拟网络 VS. 面向容器应用的虚拟网络

TCP/IP Stack

Application Layer (HTTP, FTP, etc.)

Transport Layer (TCP，UDP, SCTP, etc.)

Internet Layer (IPv4, IPv6)

Link Layer (Ethernet, etc.)

VM — APP APP — TCP/IP X — vNIC

VM — APP APP — TCP/IP Y — vNIC

VM — APP APP — TCP/IP Z — vNIC

Virtual Networking (vSwitch)

APP APP APP APP APP

Virtual Networking

| 面向虚拟机的虚拟网络 | | 面向容器应用的虚拟网络 |
|---|---|---|
| 1. 为虚拟机提供网络连接，看不到应用 | ➡ | 1. 为应用（基于容器）提供连接能力 |
| 2. 应用部署于虚拟机内 | ➡ | 2. 应用部署于单个容器，或者多个容器 |
| 3. 虚拟网络基于虚拟交换构建 | ➡ | 3. 虚拟网络涵盖了TCP/IP全栈 |
| 4. 安装应用是可以选择OS和网络协议栈 | ➡ | 4. 应用根据需要，定制化部署的协议栈 |
| 5. 部署：继承了物理数据中心的部署方式，网络需要根据应用提前规划，然后与应用割裂 | ➡ | 5. 部署：网络由应用定义，组建和销毁，网络提供连通性和网络功能的SLA |

## 应用定义网络五大特征：

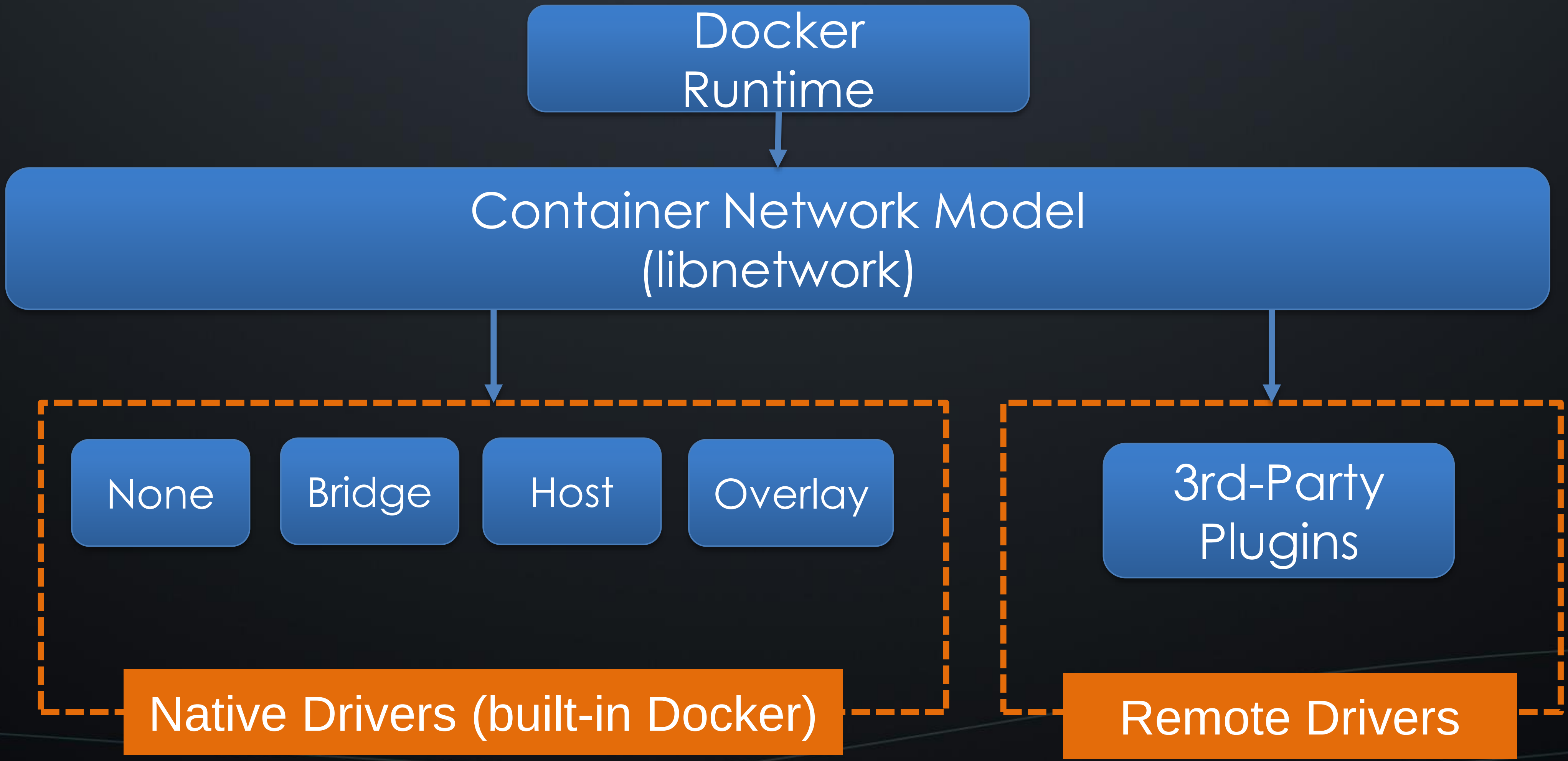
1. 根据应用部署蓝图，按需提供虚拟网络组网
2. 提供 “应用内” 和 “应用间” 高质量通讯
3. 提供可定制的应用网络状态监控和故障诊断
4. 提供应用可定义的网络SLA能力
5. 针对不同的应用按需提供定制的网络能力

# Container Network Specifications

There are two proposed standards for configuring network interfaces for Linux Containers

- Container Network Model：Docker 提出的规范
- Container Network Interface：CoreOS提出的一个容器网络规范。已采纳该规范的包括Apache Mesos, Cloud Foundry, Kubernetes, Kurma 和 rkt。

# Container Network Model (CNM) Drivers



Docker Runtime

Container Network Model (libnetwork)

None

Bridge

Host

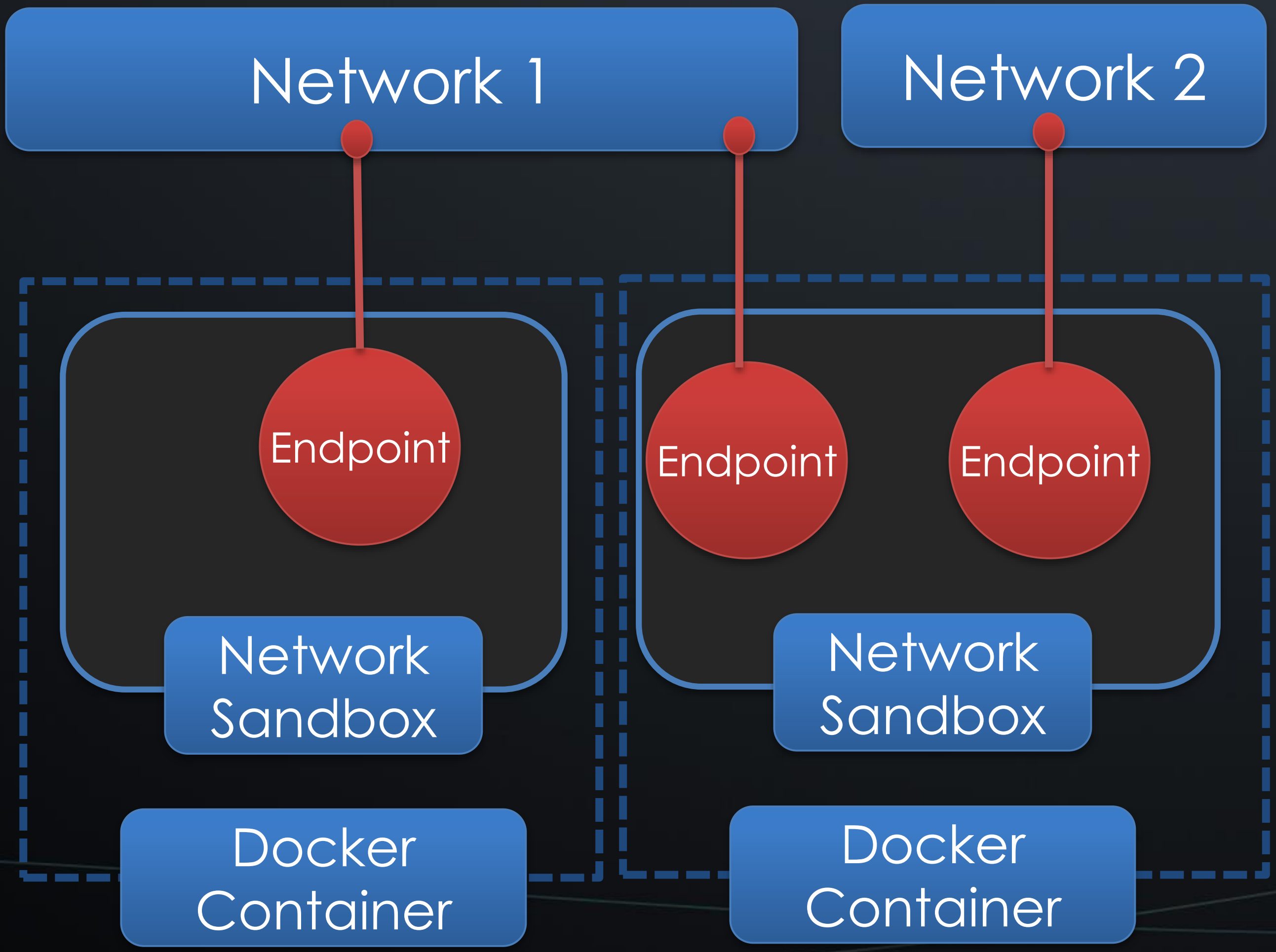
Overlay

Native Drivers (built-in Docker)

3rd-Party Plugins

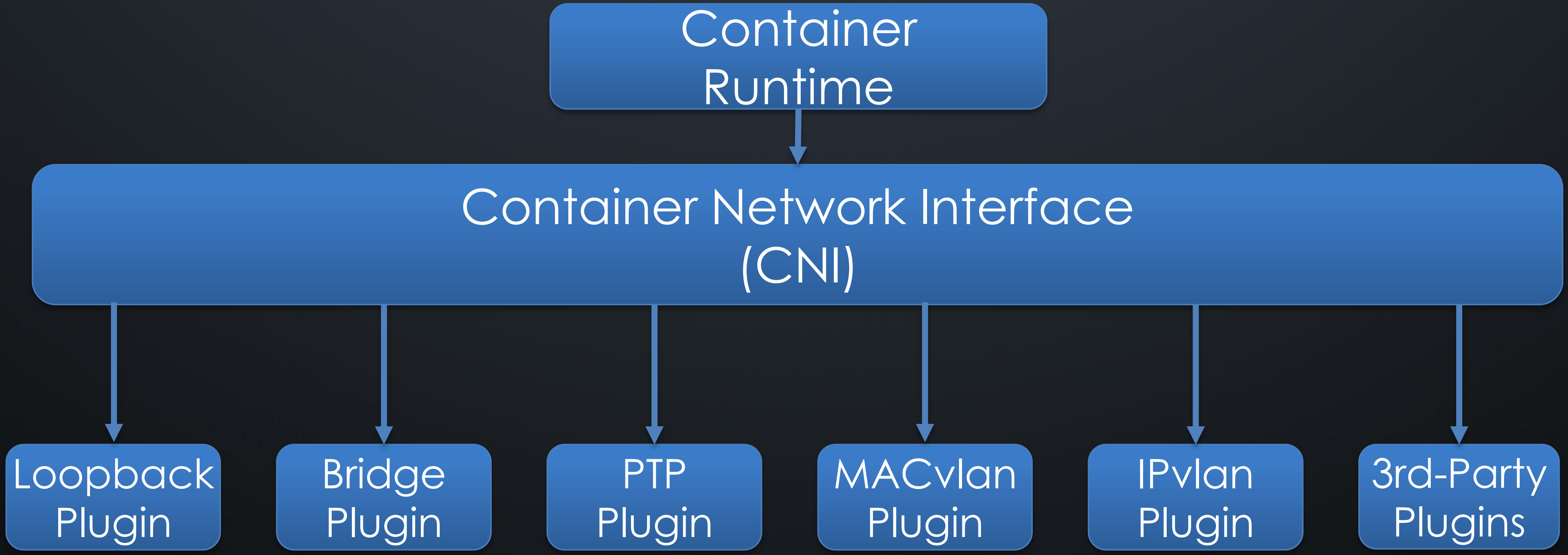
Remote Drivers

# Container Network Model

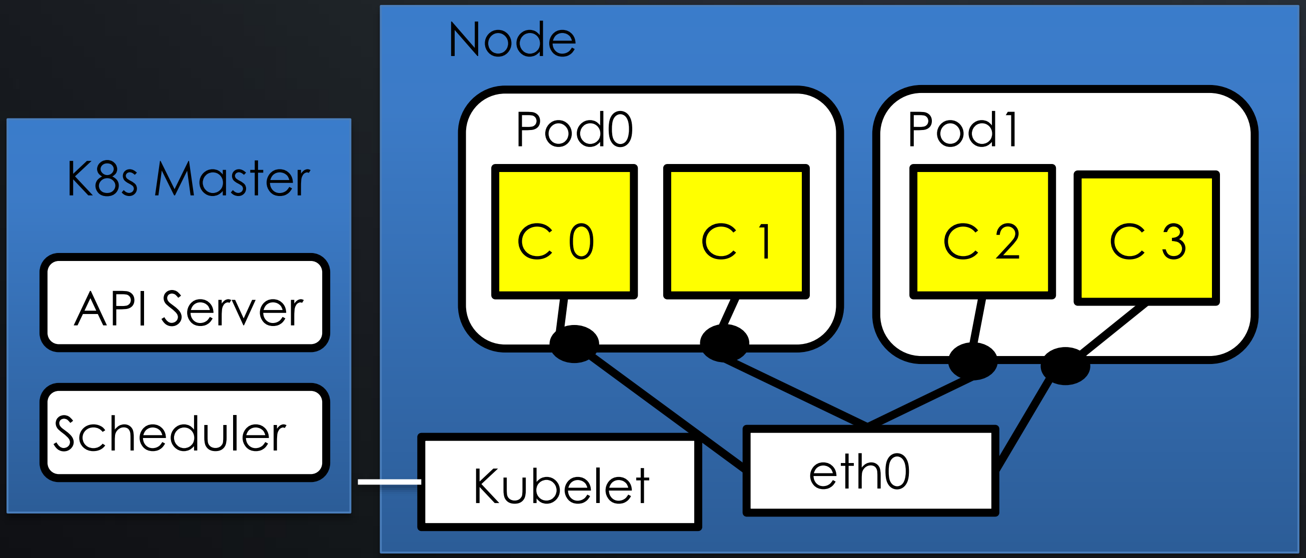


- **CNM**
  - ✓ Sandbox : Network Stack in the Container
  - ✓ Endpoint : Paired Interface between Sandbox and Network
  - ✓ Network : External Network
  - ✓ Native CNM implemented by Libnetwork , supports none, bridge, host, overlay and Underlay
  - ✓ Remote Driver can support third part driver plug-in

# Container Network Interface(CNI) Drivers

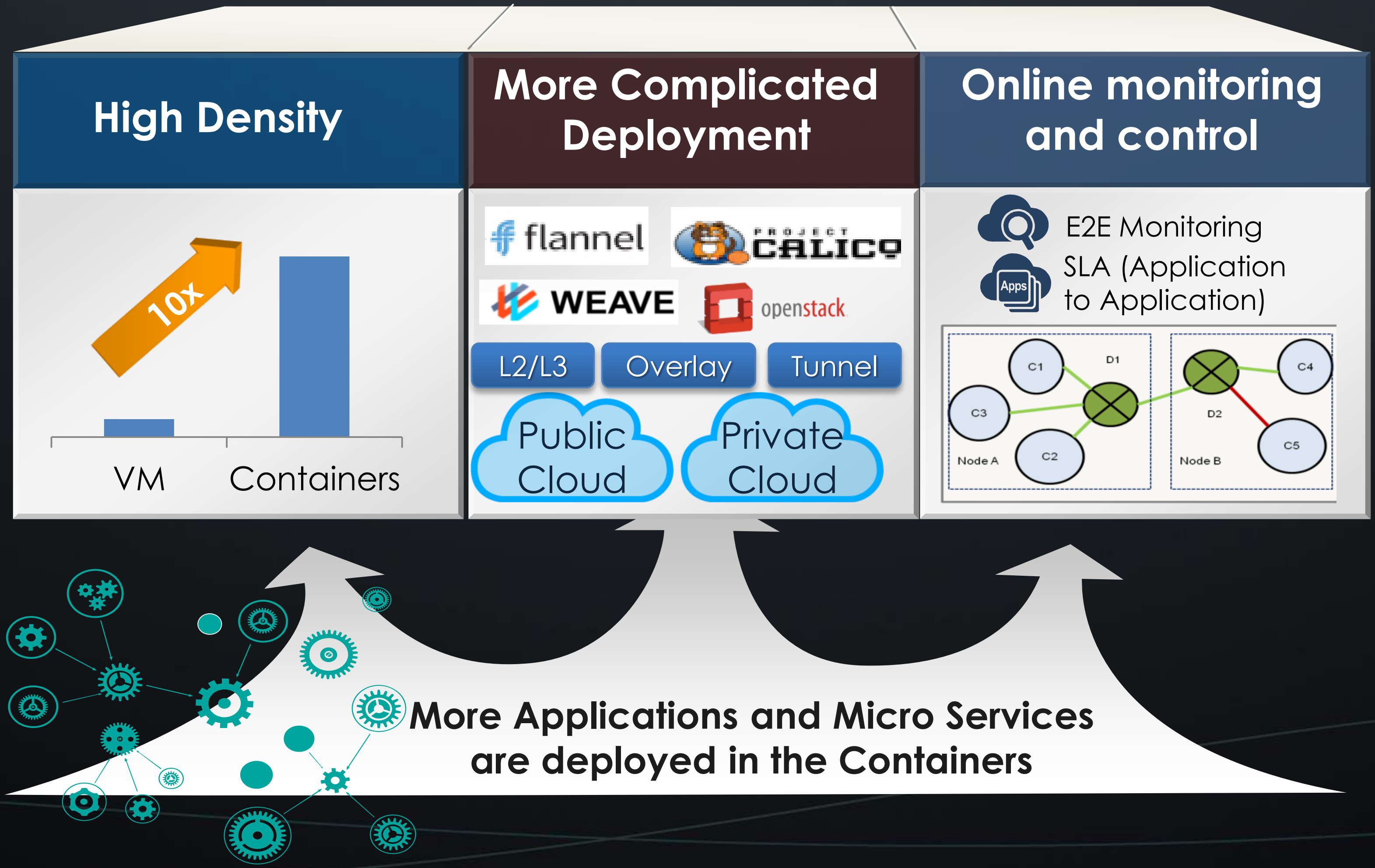


# Container Network Interface



- **CNI**
  - ✓ Network Configure : defined by Jason;
  - ✓ Interface support “Add” and ”Remove”
  - ✓ A CNI plugin is implemented as an executable, responsible for wiring up the container and IPAM.
  - ✓ Support by Kubernetes

# cloud native and containerised micro-services



High Density

More Complicated Deployment

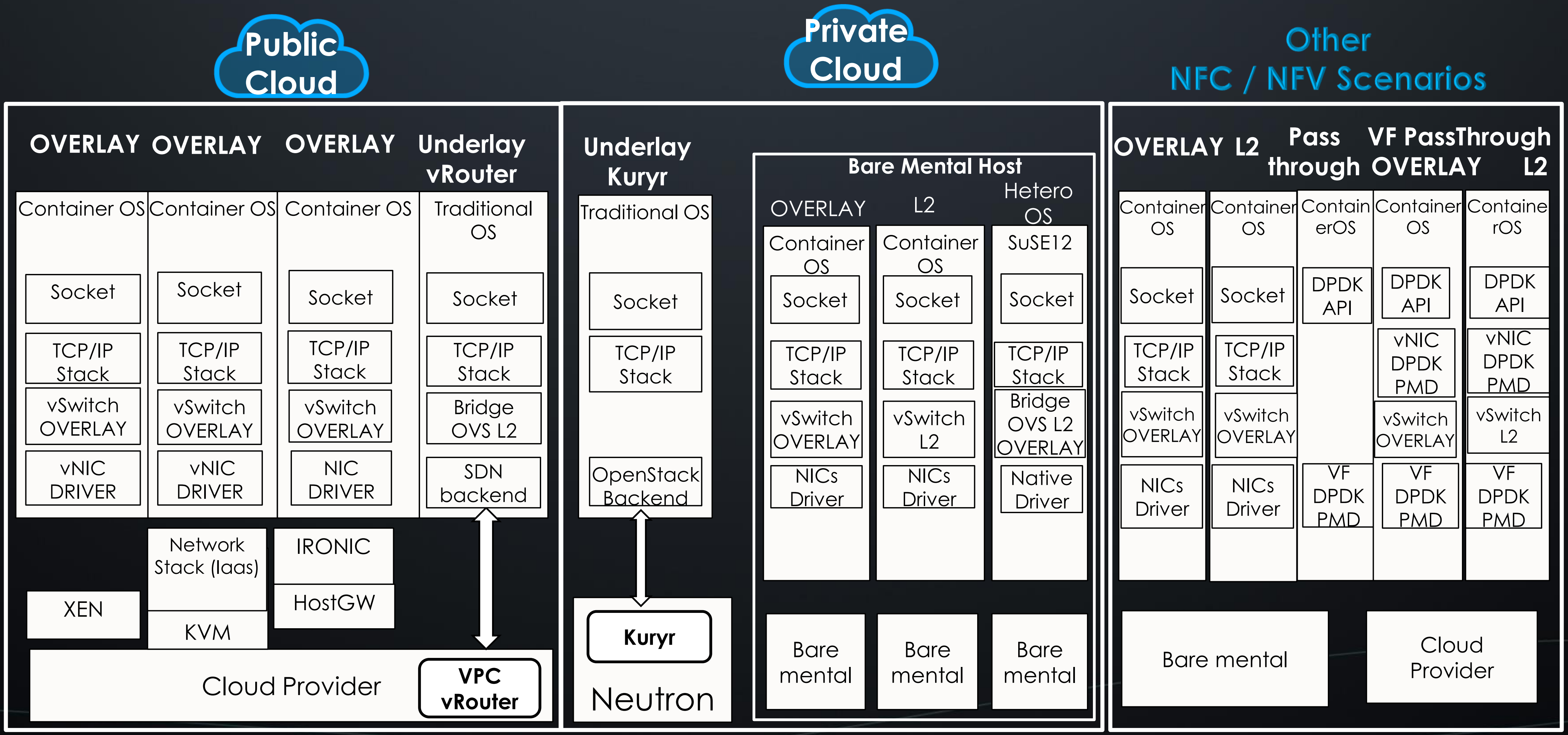
Online monitoring and control

E2E Monitoring

SLA (Application to Application)

More Applications and Micro Services are deployed in the Containers

# How we deal with so many scenarios for containers?



Public Cloud

OVERLAY: Container OS / Socket / TCP/IP Stack / vSwitch OVERLAY / vNIC DRIVER / XEN

OVERLAY: Container OS / Socket / TCP/IP Stack / vSwitch OVERLAY / vNIC DRIVER / Network Stack (Iaas) / KVM

OVERLAY: Container OS / Socket / TCP/IP Stack / vSwitch OVERLAY / NIC DRIVER / IRONIC / HostGW

Underlay vRouter: Traditional OS / Socket / TCP/IP Stack / Bridge OVS L2 / SDN backend ↕ VPC vRouter

Cloud Provider

Private Cloud

Underlay Kuryr: Traditional OS / Socket / TCP/IP Stack / OpenStack Backend ↕ Kuryr

Neutron

Bare Mental Host

OVERLAY: Container OS / Socket / TCP/IP Stack / vSwitch OVERLAY / NICs Driver / Bare mental

L2: Container OS / Socket / TCP/IP Stack / vSwitch L2 / NICs Driver / Bare mental

Hetero OS: SuSE12 / Socket / TCP/IP Stack / Bridge OVS L2 OVERLAY / Native Driver / Bare mental

Other NFC / NFV Scenarios

OVERLAY: Container OS / Socket / TCP/IP Stack / vSwitch OVERLAY / NICs Driver

L2: Container OS / Socket / TCP/IP Stack / vSwitch OVERLAY / NICs Driver

Pass through: ContainerOS / DPDK API / VF DPDK PMD

VF PassThrough OVERLAY: Container OS / DPDK API / vNIC DPDK PMD / vSwitch OVERLAY / VF DPDK PMD

L2: ContainerOS / DPDK API / vNIC DPDK PMD / vSwitch L2 / VF DPDK PMD

Bare mental

Cloud Provider

# Deployment Complexity

simple flat container network model: CNI

complex deployment scenarios

public clouds: AWS/Azure/HEC

private clouds: openstack/vmware/ baremetal

NFV: SR-IOV/L2/L3

# Deployment Complexity

simple flat container etwork model: CNI

existing solutions are suitable for limited cases with hard-coded "plugins"

flannel
PROJECT CALICO
WEAVE
Contiv
openstack.

mplex deployment scenarios

public clouds: AWS/Azure/HEC

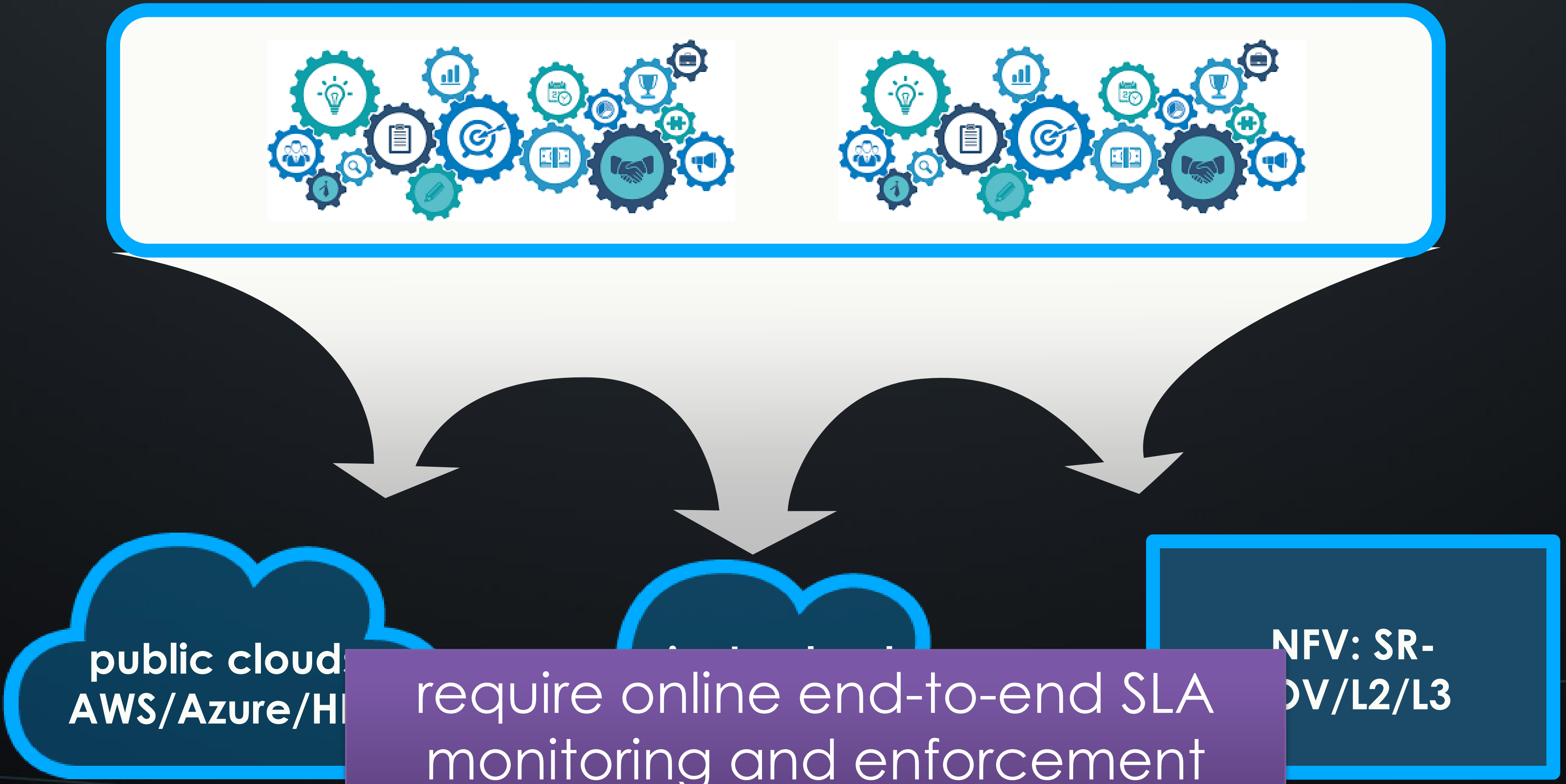
private clouds: openstack/vmware/ baremetal

NFV: SR-IOV/L2/L3

require a flexible solution that always adapts the best technology based on specific situation

# Online monitoring and control

## various deployments may yield different performance



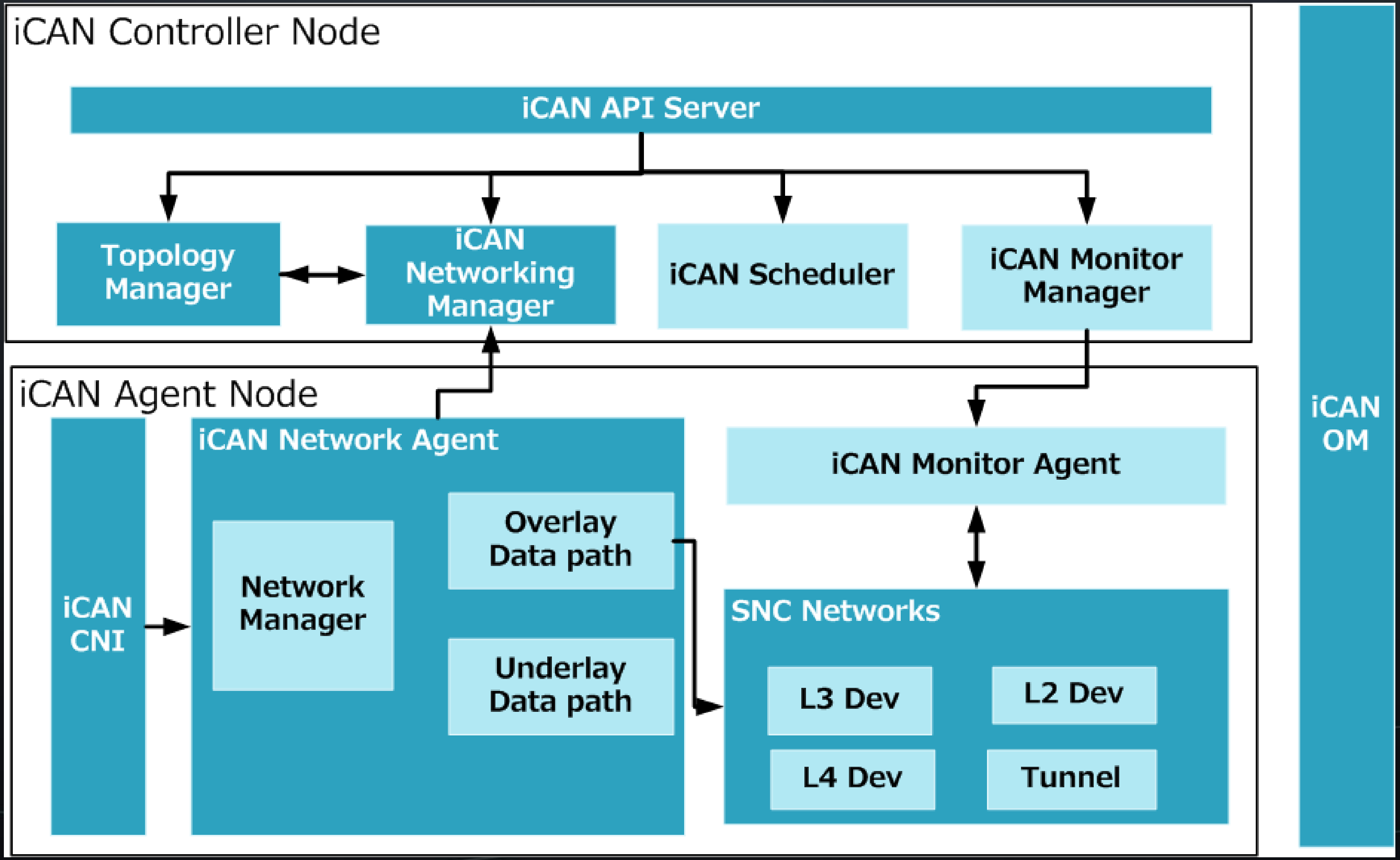
require online end-to-end SLA monitoring and enforcement

# Our Solution: iCAN (intelligent Container Network)

an extensible framework to

- program various container network data path and policies
- adapt to different orchestrators
- support end-to-end SLA between containerised applications

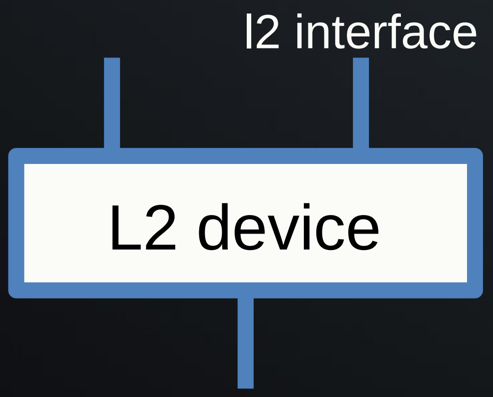
# iCAN architecture



# Standard Network Component (SNC) model

## abstract for network components in data-path

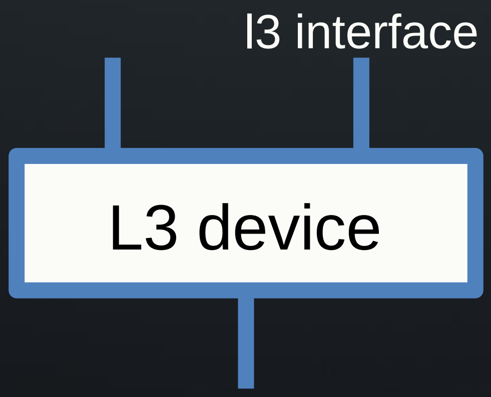
- interfaces, devices and templates



l2 interface

L2 device

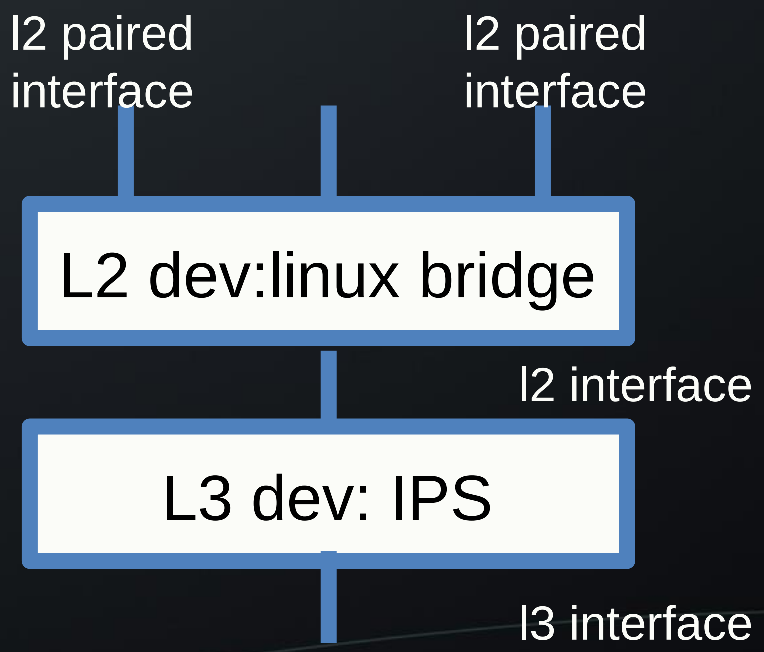
L2 devices:
bridge/macvlan/ovs/…



l3 interface

L3 device

L3 devices:
router/ipvlan/…



l2 paired interface

l2 paired interface

L2 dev:linux bridge

l2 interface

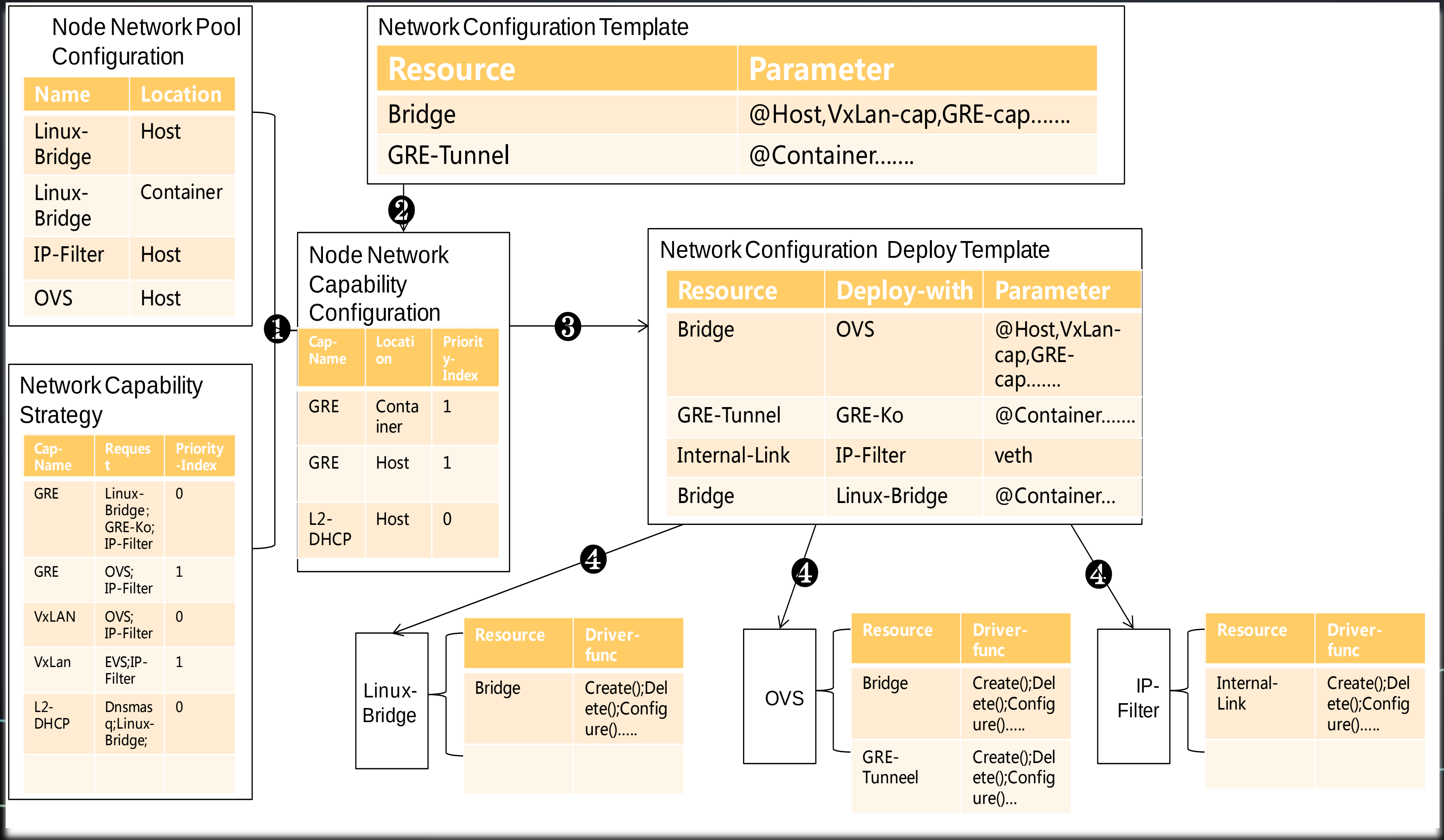
L3 dev: IPS

l3 interface

a template for Flannel
data path

# Selection of right SNC template

iCAN master emulates all possible SNC templates based on network capabilities of nodes optimally selects SNC configurations for all nodes based on SLA policies



Node Network Pool Configuration

| Name | Location |
|---|---|
| Linux-Bridge | Host |
| Linux-Bridge | Container |
| IP-Filter | Host |
| OVS | Host |

Network Capability Strategy

| Cap-Name | Request | Priority-Index |
|---|---|---|
| GRE | Linux-Bridge;<br>GRE-Ko;<br>IP-Filter | 0 |
| GRE | OVS;<br>IP-Filter | 1 |
| VxLAN | OVS;<br>IP-Filter | 0 |
| VxLan | EVS;IP-Filter | 1 |
| L2-DHCP | Dnsmasq;Linux-Bridge; | 0 |
| | | |

Network Configuration Template

| Resource | Parameter |
|---|---|
| Bridge | @Host,VxLan-cap,GRE-cap....... |
| GRE-Tunnel | @Container....... |

Node Network Capability Configuration

| Cap-Name | Location | Priority-Index |
|---|---|---|
| GRE | Container | 1 |
| GRE | Host | 1 |
| L2-DHCP | Host | 0 |

Network Configuration Deploy Template

| Resource | Deploy-with | Parameter |
|---|---|---|
| Bridge | OVS | @Host,VxLan-cap,GRE-cap....... |
| GRE-Tunnel | GRE-Ko | @Container....... |
| Internal-Link | IP-Filter | veth |
| Bridge | Linux-Bridge | @Container... |

Linux-Bridge

| Resource | Driver-func |
|---|---|
| Bridge | Create();Delete();Configure()..... |
| | |

OVS

| Resource | Driver-func |
|---|---|
| Bridge | Create();Delete();Configure()..... |
| GRE-Tunneel | Create();Delete();Configure()... |

IP-Filter

| Resource | Driver-func |
|---|---|
| Internal-Link | Create();Delete();Configure()..... |
| | |

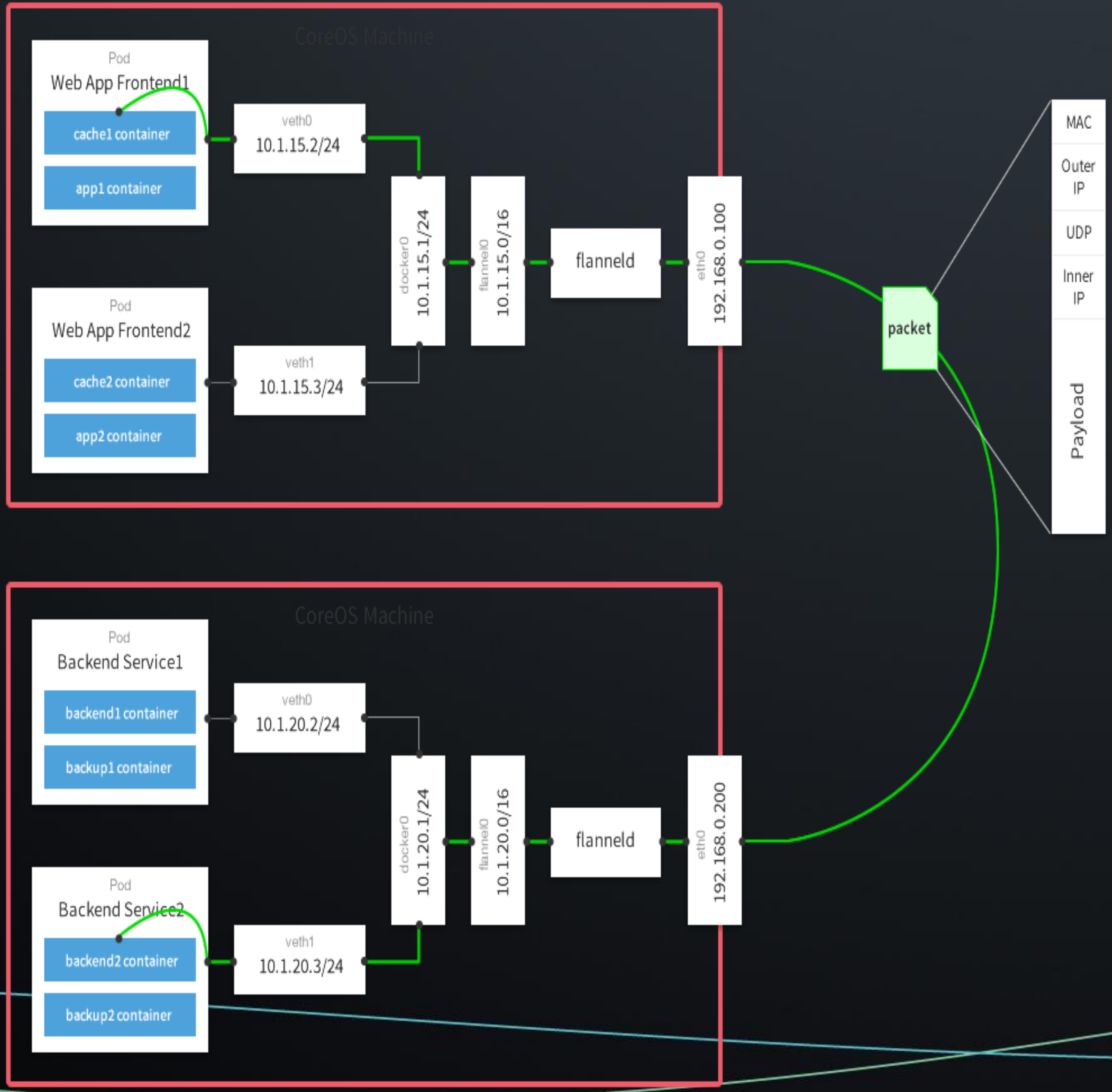
❶ Network-Agent Local 初始化，根据本地资源池（Node Network Pool Configuration）及网络能力策略库（Network Capability Strategy），综合得出节点网络能力配置部署能力表（Node Network Capability Configuration 维护）；

❷ Node接到模板部署请求，送入本地网络能力模块，如无法满足直接失败返回，否则输出带Deploy-With信息的部署模板❸；

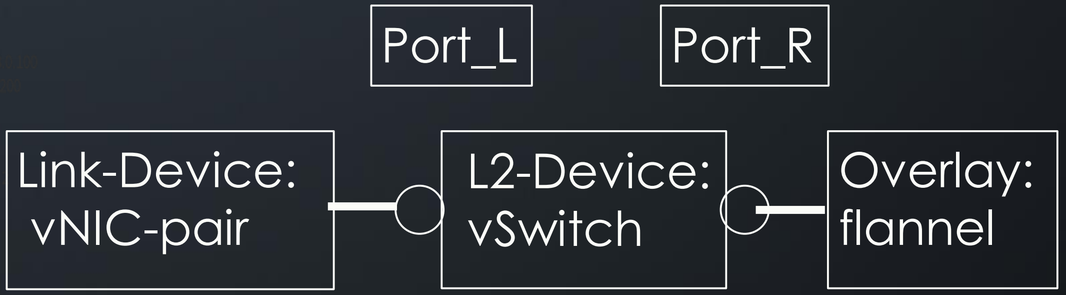
❹而后，依照部署模板信息，依次将各个网络资源部署请求送入相应的Network-Element处理单元，由相应驱动负责最终落地；

注意：在❸过程中，需要同时生成AccessEndpoint及容器内资源部署信息，作为容器Join进入Network过程指示；

# example: Flannel with SNC



## Flannel SNC template:



Port_L

Port_R

Link-Device: vNIC-pair

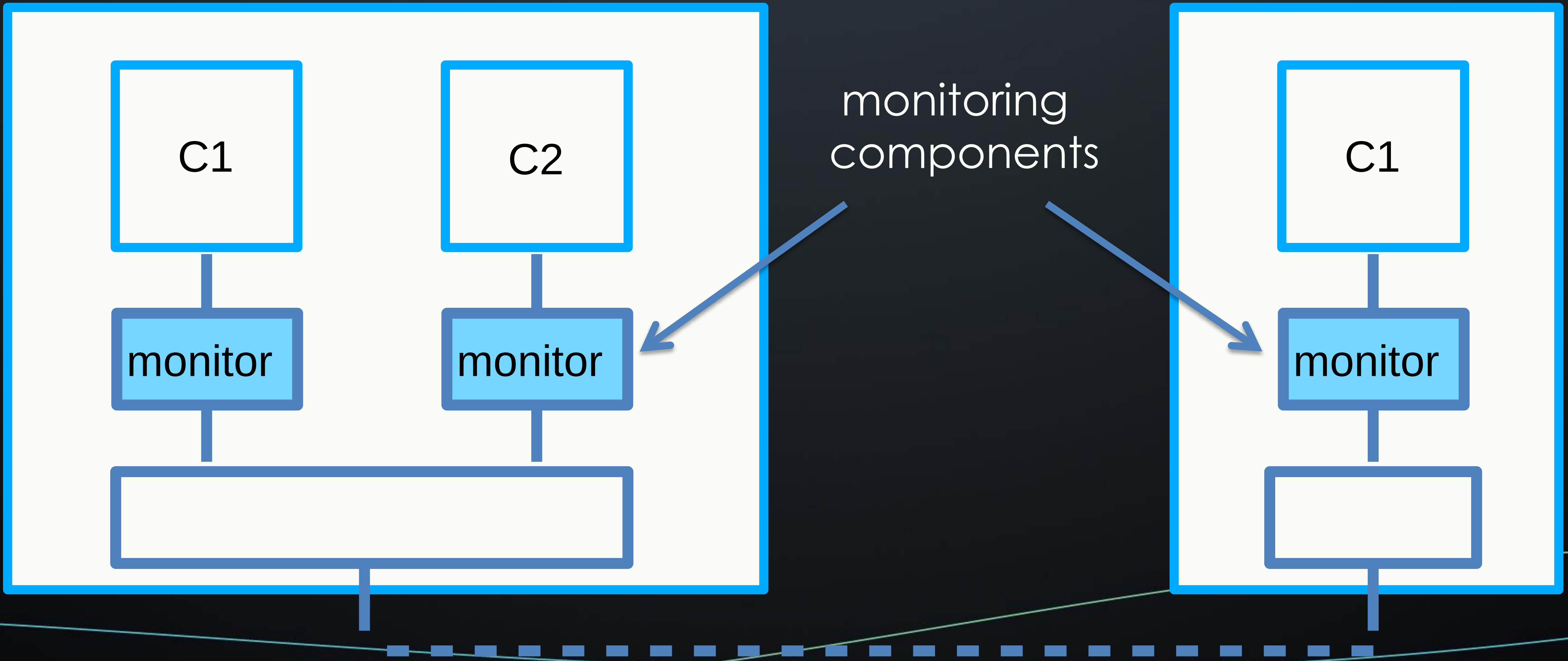
L2-Device: vSwitch

Overlay: flannel

template json:

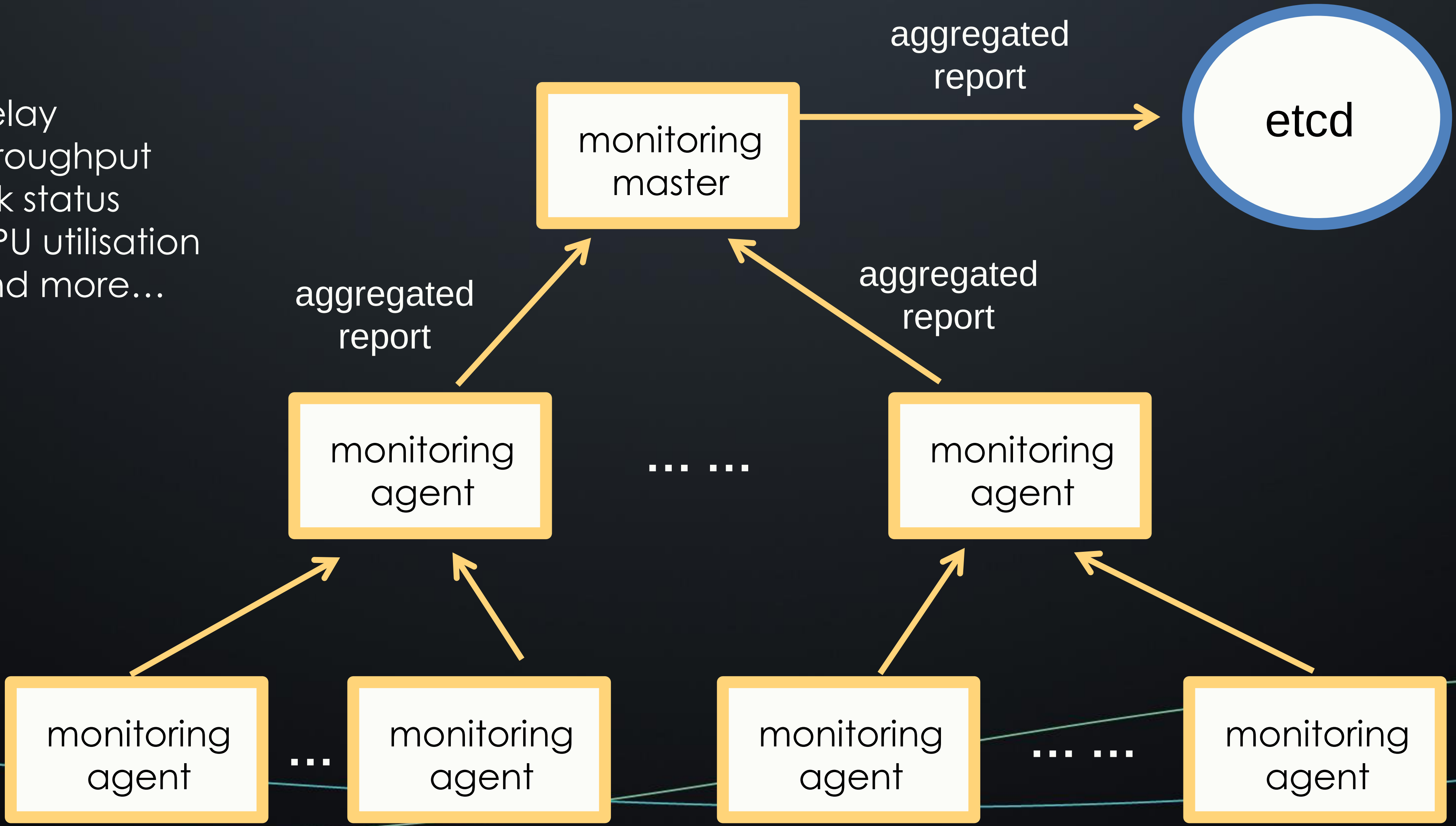
```
main-network: {
  node: [ {
       name:  “br-int”,
       type: [“bridge”,  “ovs”],
       link-point : […]
      },
      { name:  “br-tunnel”,
        type:  “flannel-udp”,
        …
      } ]
```

# monitoring components



# Monitoring report aggregation

- ✓ delay
- ✓ throughput
- ✓ link status
- ✓ CPU utilisation
- ✓ and more…

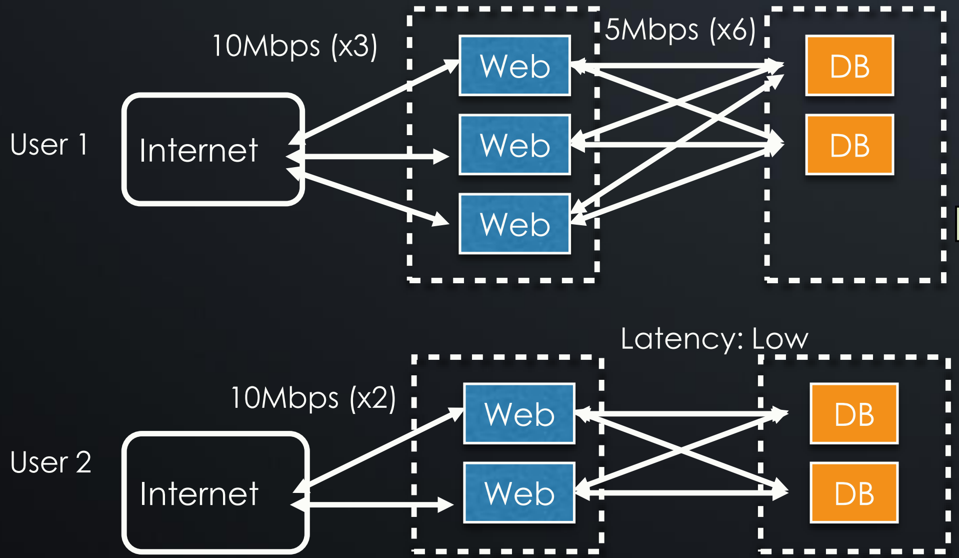


# Summary of iCAN monitoring components

| metrics | data source |
|---|---|
| **E2E Latency** | Provide UDP,TCP,ICMP based one way and two ways detection |
| **E2E Bandwidth** | Average single point data in central |
| **E2E PKT Loss Rate** | Compare single point data in central |
| **Traffic Analysis** | IP stack statistic program for local Pods<br>Multiple steps efforts for cross hosts |

| metrics | data source |
|---|---|
| **Bandwidth Capacity** | •Between vNIC and pNIC, maximum is pNic Speed<br>•Between vNic, no fixed upper limitation. Can calculate in static mode |
| **Current Bandwidth** | Single point interface RX/TX packets , bytes |
| **Runtime Status** | Single point interface RX/TX errors, dropped, overrun |
| **Traffic Analysis** | Traffic filter (collecting through enable all vPorts) |

# Simplify Network SLA modeling



User 1

Internet

10Mbps (x3)

Web

Web

Web

5Mbps (x6)

DB

DB

Latency: Low

User 2

Internet

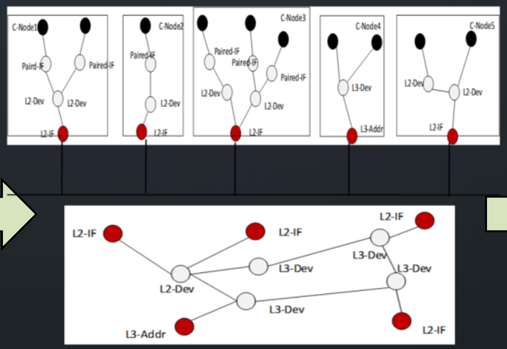
10Mbps (x2)

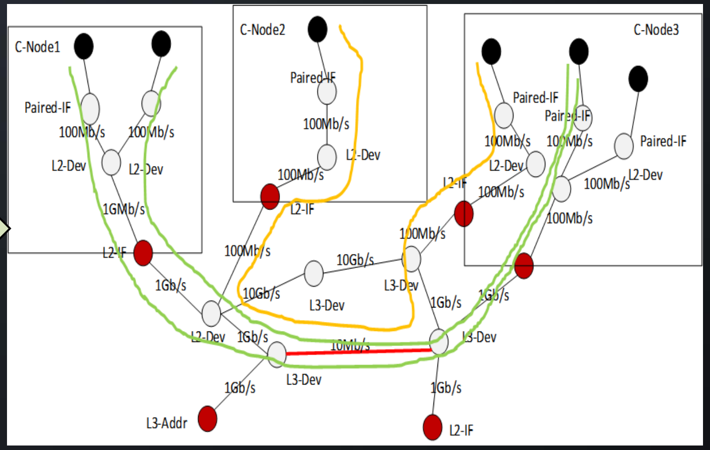
Web

Web

DB

DB





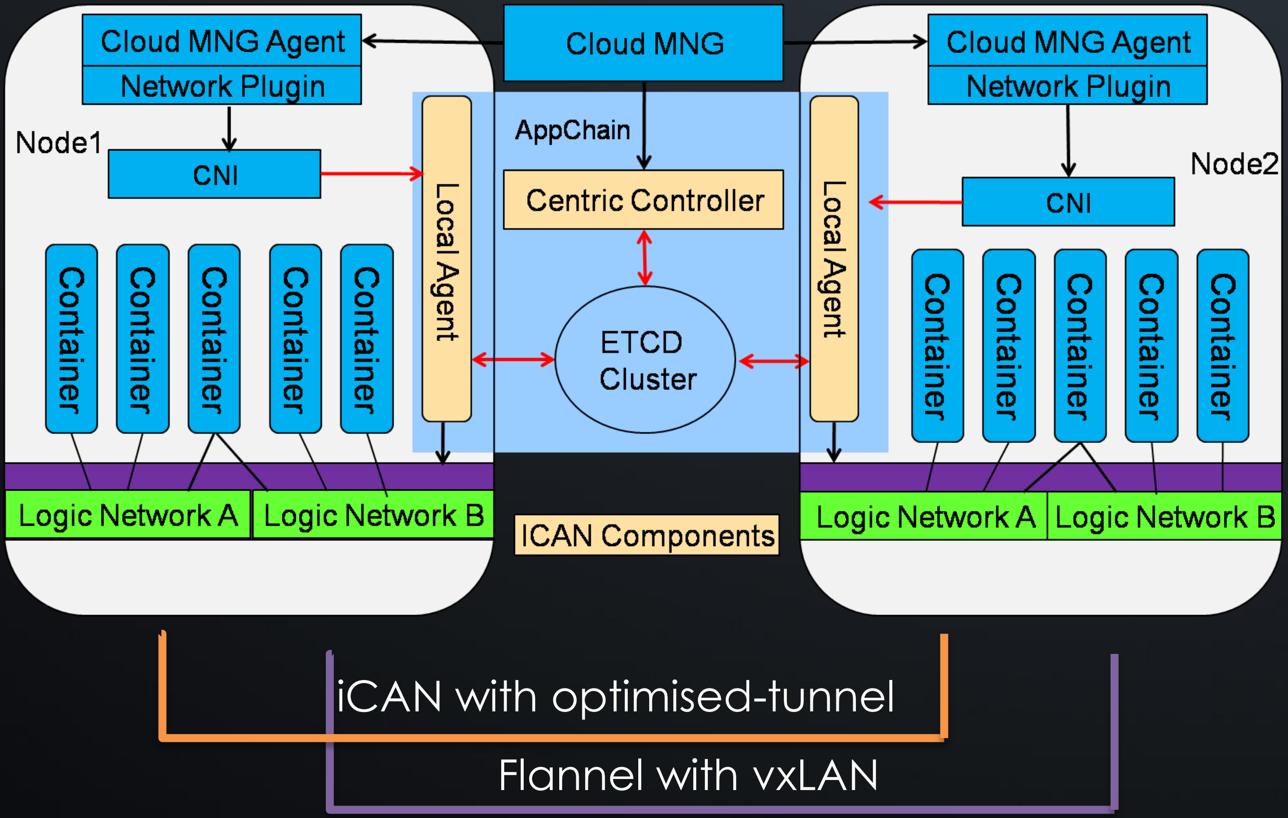
convert link requirement to node requirement

SLA-annotated polices

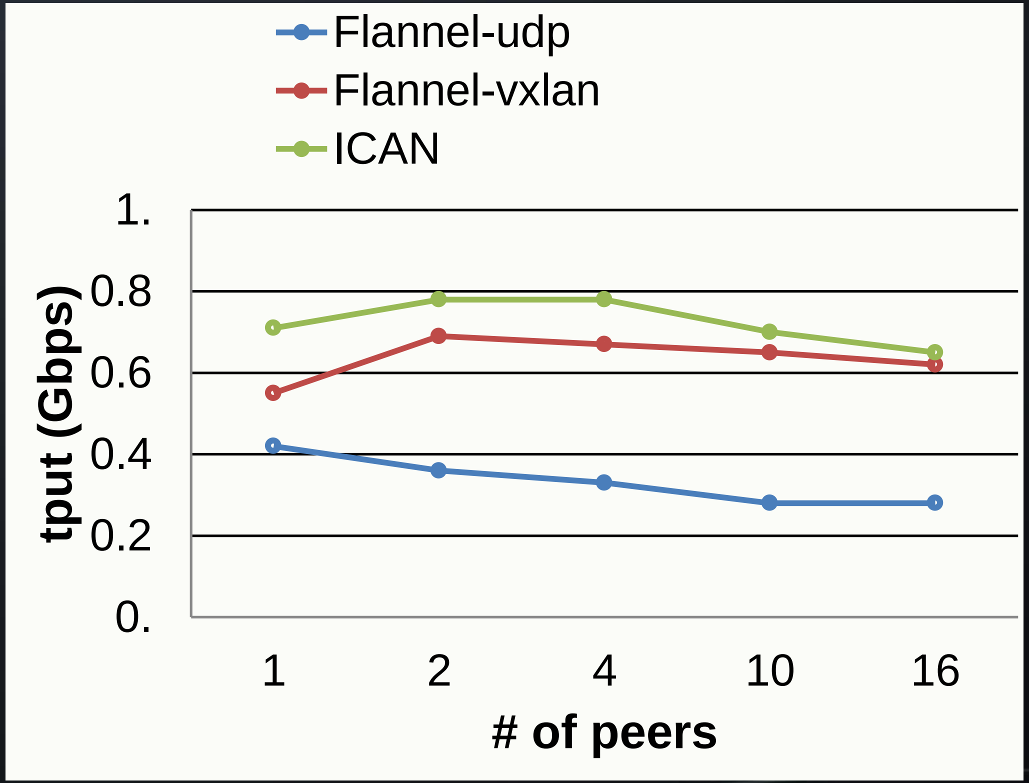
monitoring

SLA feedback & rescheduling

# case study



Cloud MNG
Cloud MNG Agent
Network Plugin
Node1
CNI
AppChain
Centric Controller
Local Agent
ETCD Cluster
Local Agent
Cloud MNG Agent
Network Plugin
Node2
CNI
Container
Container
Container
Container
Container
Container
Container
Container
Container
Container
Logic Network A
Logic Network B
ICAN Components
Logic Network A
Logic Network B
iCAN with optimised-tunnel
Flannel with vxLAN



Flannel-udp
Flannel-vxlan
ICAN
tput (Gbps)
1.
0.8
0.6
0.4
0.2
0.
1
2
4
10
16
# of peers

# iCAN summary

flexible and extensible framework for diverse deployment usages
using SNC model

integrated monitoring capability for container networks
network SLA specification and end-node based enforcement

# thanks! questions?

3rd NJSD

Global Software Development Conference . Nanjing

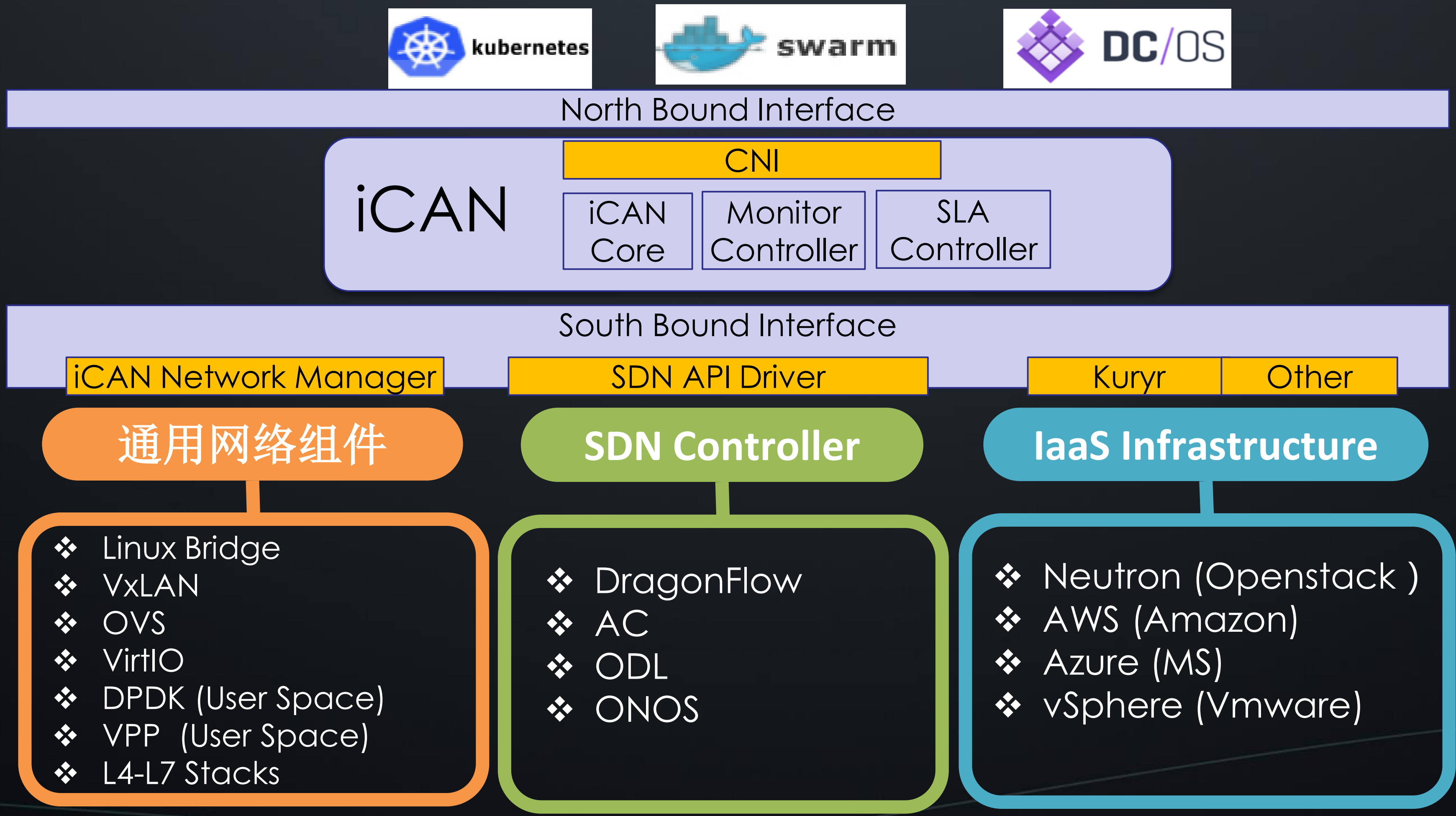
全球软件大会

2017

# Thank You.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.

# iCAN Community Strategy



kubernetes | swarm | DC/OS

North Bound Interface

iCAN

CNI

iCAN Core | Monitor Controller | SLA Controller

South Bound Interface

iCAN Network Manager | SDN API Driver | Kuryr | Other

**通用网络组件**

- Linux Bridge
- VxLAN
- OVS
- VirtIO
- DPDK (User Space)
- VPP  (User Space)
- L4-L7 Stacks

**SDN Controller**

- DragonFlow
- AC
- ODL
- ONOS

**IaaS Infrastructure**

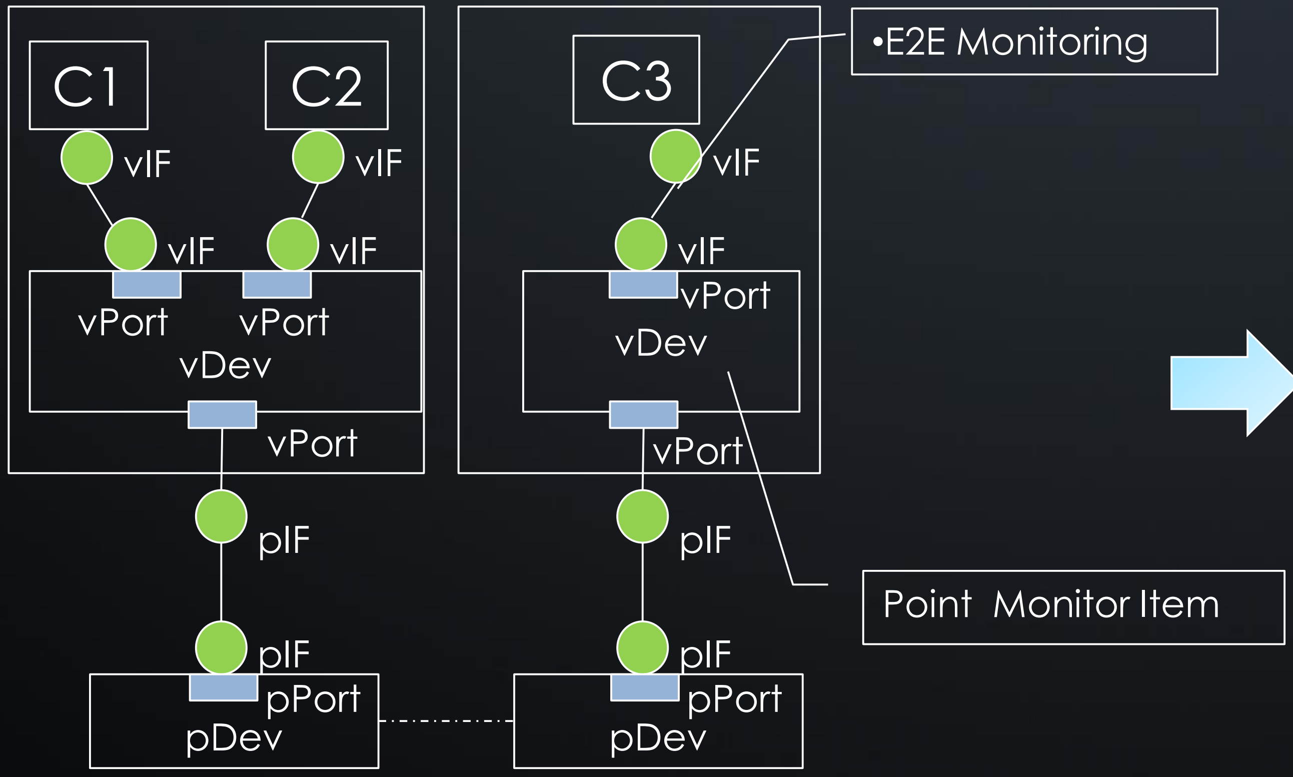
- Neutron (Openstack )
- AWS (Amazon)
- Azure (MS)
- vSphere (Vmware)

# Existing Container Network Solutions

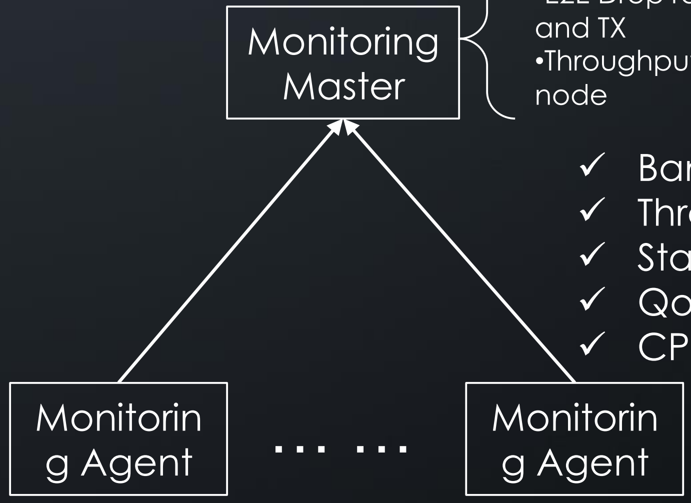
| Solution Comparison | Weave<br>WEAVE | Flannel<br>flannel | Contiv on ACI | Kuryr@Neutron<br>openstack | Calico<br>PROJECT CALICO |
|---|---|---|---|---|---|
| **Basic Networking** | VXLAN or UDP Overlay | VXLAN or UDP Overlay | L2, L3(BGP)<br>VXLAN Overlay | L2 via vSwitch | L3(BGP) |
| **Optimized stack for App** | Private UDP Tunnel | VXLAN+ Private Tunnel | No | No | Linux IP +BGP |
| **Application Isolation** | CIDR | CIDR | Tent isolation<br>Policy based Label | Rely on Neutron | Policy based on Label, Port , CIDR |
| **Monitoring** | No | No | Just monitor in the physical network | No | No |
| **Network SLA** | No | No | QoS via EPG; no SLA for App | No | No |
| **CNI** | Yes | Yes | Yes | Yes | Yes |
| **CNM** | Yes | No | Yes | No | Yes |
| **Security** | Encrypt Channel | No | Support firewall | Depend on IaaS | Rely Linux Capabilities |
| **Preferred Scenario** | Less nodes, Simple L3 Network | Complicated environment, Multi-subnets | Multi-Tent<br>Public cloud | Openstack<br>Public cloud<br>Private Cloud | Cross DC |

# Monitoring based SNC Modeling

Monitoring on local SNC components :



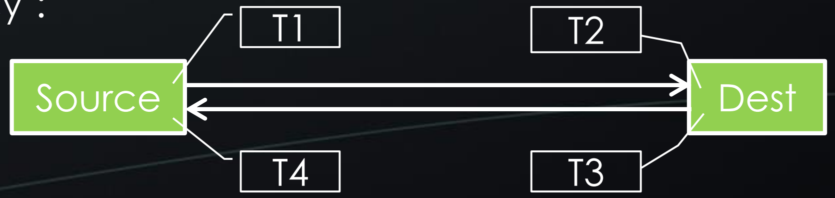
Generate E2E monitoring data in master node :



- •E2Ethrought：minimal throughput
- •E2E Drop rate： deviations between RX and TX
- •Throughput Analysis：data from local node

✓ Bandwidth
✓ Throughput
✓ Status
✓ QoS
✓ CPU utilization

Latency :



Latency = ((T4 - T1) - (T3 - T2)) / 2

# Monitoring Bases Modeling Network Node

Monitoring Usage:

| | | |
|---|---|---|
| SLA Monitoring | Network Performance View | Network Topology View |

End to End Monitoring in Master Node:

| | | | | | |
|---|---|---|---|---|---|
| Pod to Pod | Pod to vNic | vNic to vNic | vNic to pNic | pNic to pNic | Tunnel |

Point Monitoring in Agent Node:

| | | | | |
|---|---|---|---|---|
| Virtual Interfaces | Virtual Ports | Virtual Network Device | Physical NIC | Physical Network Device |

| E2E Monitoring | Monitoring Data Source |
|---|---|
| **E2E Latency** | Provide UDP,TCP,ICMP based one way and two ways detection |
| **E2E Bandwidth** | Average single point data in central |
| **E2E PKT Loss Rate** | Compare single point data in central |
| **Traffic Analysis** | IP stack statistic program for local Pods<br>Multiple steps efforts for cross hosts |

| Point Monitor Item | Monitoring Data Source |
|---|---|
| **Bandwidth Capacity** | •Between vNIC and pNIC, maximum is pNic Speed<br>•Between vNic, no fixed upper limitation. Can calculate in static mode |
| **Current Bandwidth** | Single point interface RX/TX packets , bytes |
| **Runtime Status** | Single point interface RX/TX errors, dropped, overrun |
| **Traffic Analysis** | Traffic filter (collecting through enable all vPorts) |