Network-as-a-Service on Bare-metal Cloud

Rakuten, Inc.
Cloud Platform Department
Tomohisa Egawa

JANOG47

Overview

Sharing our experience of new private cloud project focusing on **Platform** and **Network**

Contents (**23** minutes)

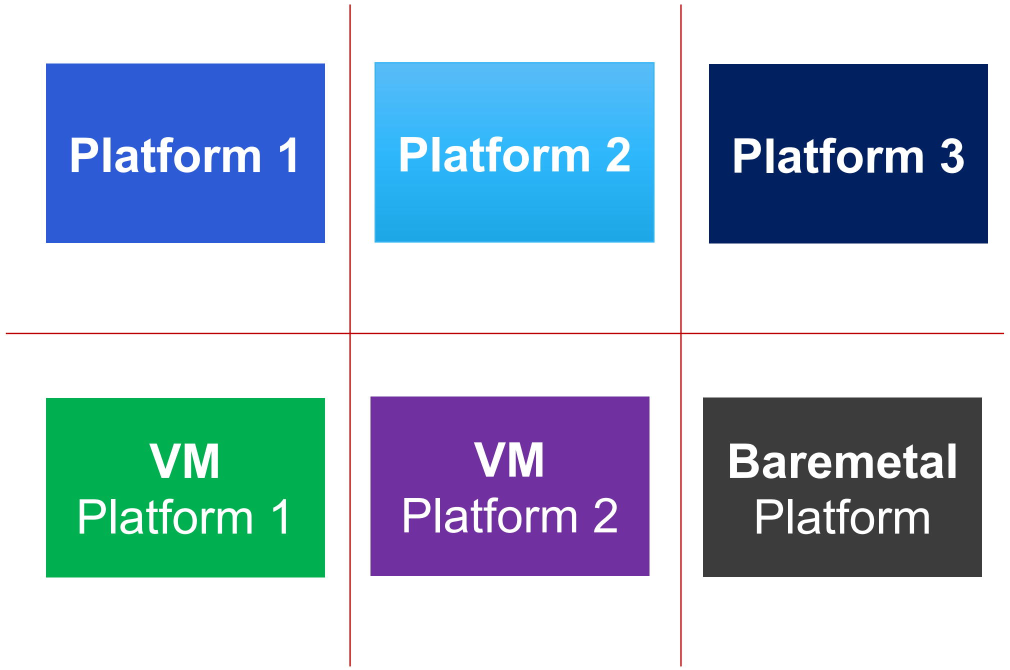
- History of our infrastructure
- Concept of new private cloud based on baremetal
- Design of platform and network

Discussion (**7** minutes)

Looking back our Infrastructure

2018: Freedom and Chaos

Silo was occurred due to platform crowd



Concerned looking ahead 3-5 years

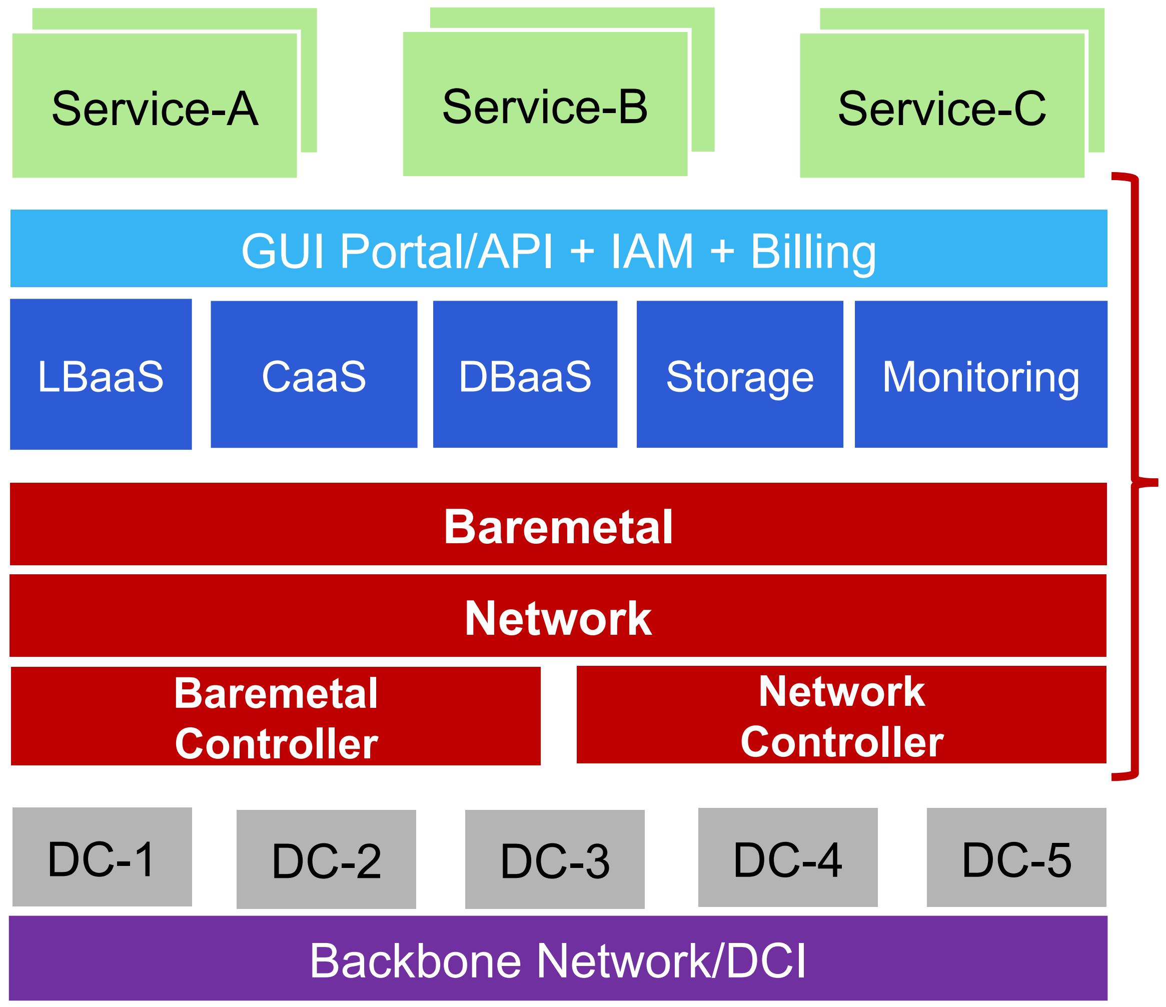
Issues

- Bad experience for in-house engineers
  - Multiple tools and portals for each platform
  - It’s difficult for users to know which one is best for their services
  - Not consolidated Billing and authentication
- Platformer view
  - Can’t keep up with life cycle even with automation
  - Upgrade cost is too heavy about platform software
  - OS management/patching to manage ton of the VMs
- Company view
  - Challenges in resource optimization at company-wide level
  - Unified security control
  - Support life cycle

Late 2018: Change Infrastructure strategy + Reorganization

New Private Cloud based on Bare-metal

Concept of Bare-metal Cloud (1)



Managed Service approach

- Reorganization with inverse Conway's law
- New services are created by combining the managed services
- Generate higher-level managed service based on the combination

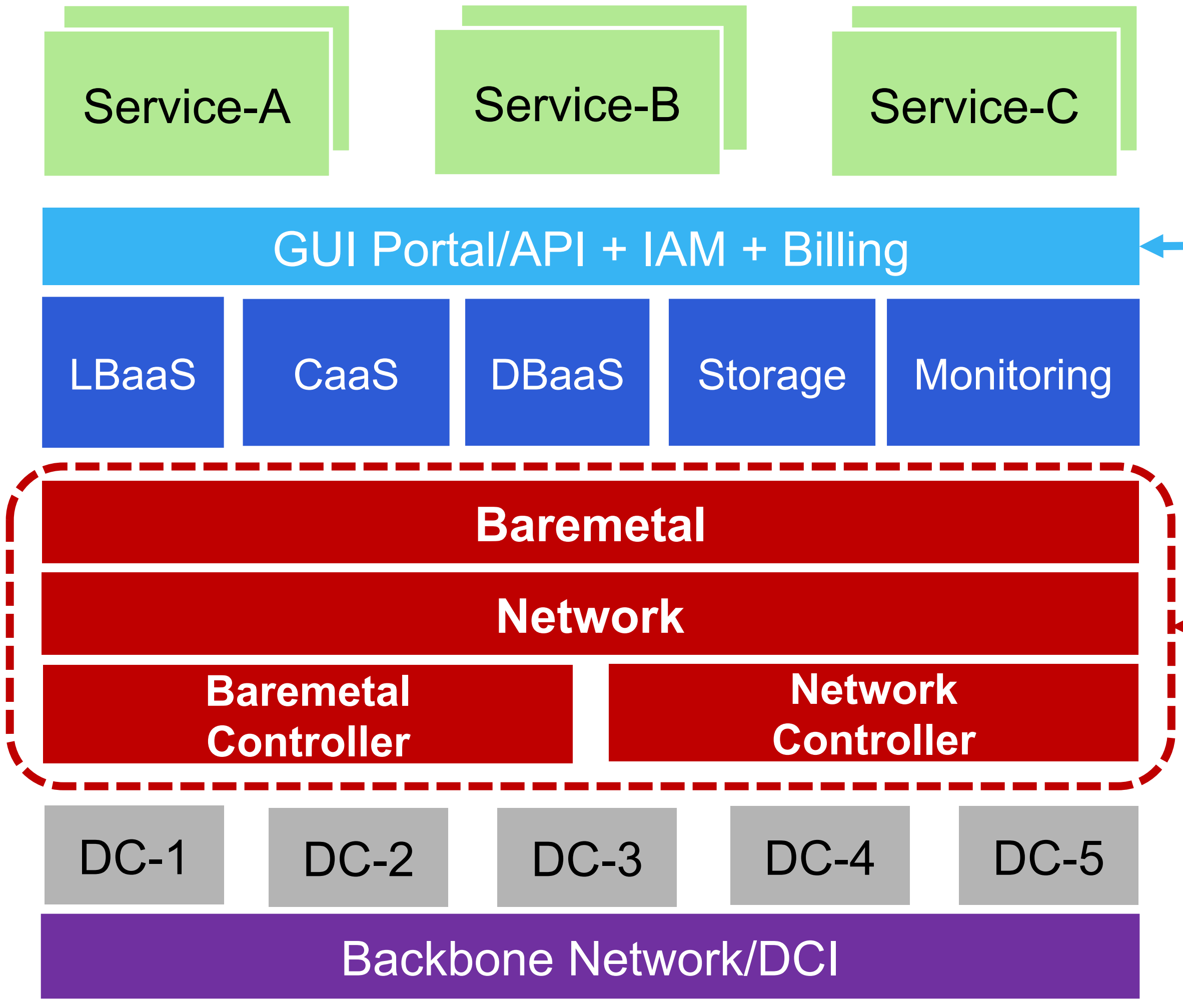
Each component is loosely coupled

- Clarification of the demarcation point of responsibility
- Update process can be performed any time
- Right technology can be selected according to the trend

Let's use CNCF/OSS product mainly

- Adopt de fact standard software
- Easy to obtain information and lower learning costs
- Cut CAPEX dramatically

Concept of Bare-metal Cloud (2)



Develop our original IAM and Portal by using OSS

- Support for frequent internal reorganizations
- Support for irregular delegation of authority

Private cloud based on bare-metal

- Bare-metal server can be provided to all in-house users
- Build a managed service from Bare-metal

Focusing on **"Core-Infrastructure"**
from the next slide

Design of Core-Infrastructure

Design concept of Core-Infrastucture

1. Sutainable Core-Infrastructure

- Being strong against the change of technology trends and having full control
- Design to keep the latest version without affecting the services on the Core-Infrastructure
- Support Multi-tenancy

2. (Stable + Scalability + Easy-operation) Network

- No risk of whole network failure by eliminating SPOF
- Ensure sufficient scalability without worrying about limitation
- Easy operation = We can keep extra capacity for the next challenge

3. Network-aaS

- Provide useful network function for users via API/GUI
- Network is also treated as “Product”

Goal 1

Sustainable Core-Infrastructure

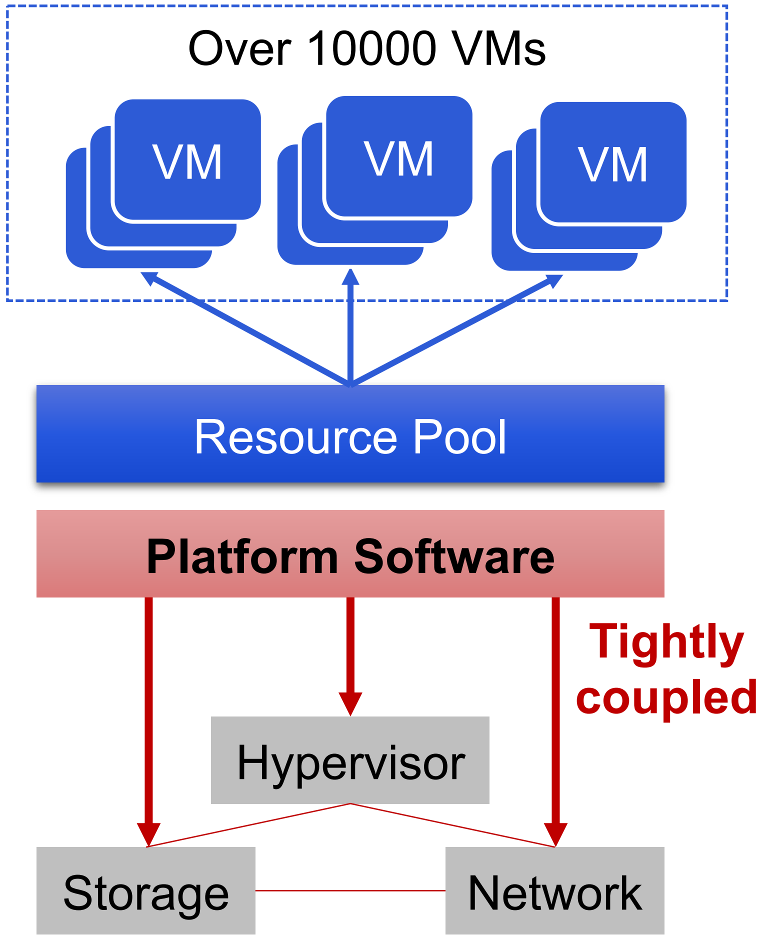
Comparison: Virtulization Infrastructure

Pros

- Integrated management through platform software
- Resource abstraction
  - Users don't need to care about physical servers
  - Optimizing utilization by pooling resources

Cons

- Platform software management is high OPEX
  - Difficult to keep up with life cycle and upgrades
  - Impact of changes is a risk against all services
- Increased OPEX due to large number of VMs
  - Configuration drift even using IaC
  - OS-level security patching



Suppress the number of VM → Increse the ratio of Container

Comparison: Core-Infrastructure based on Bare-metal

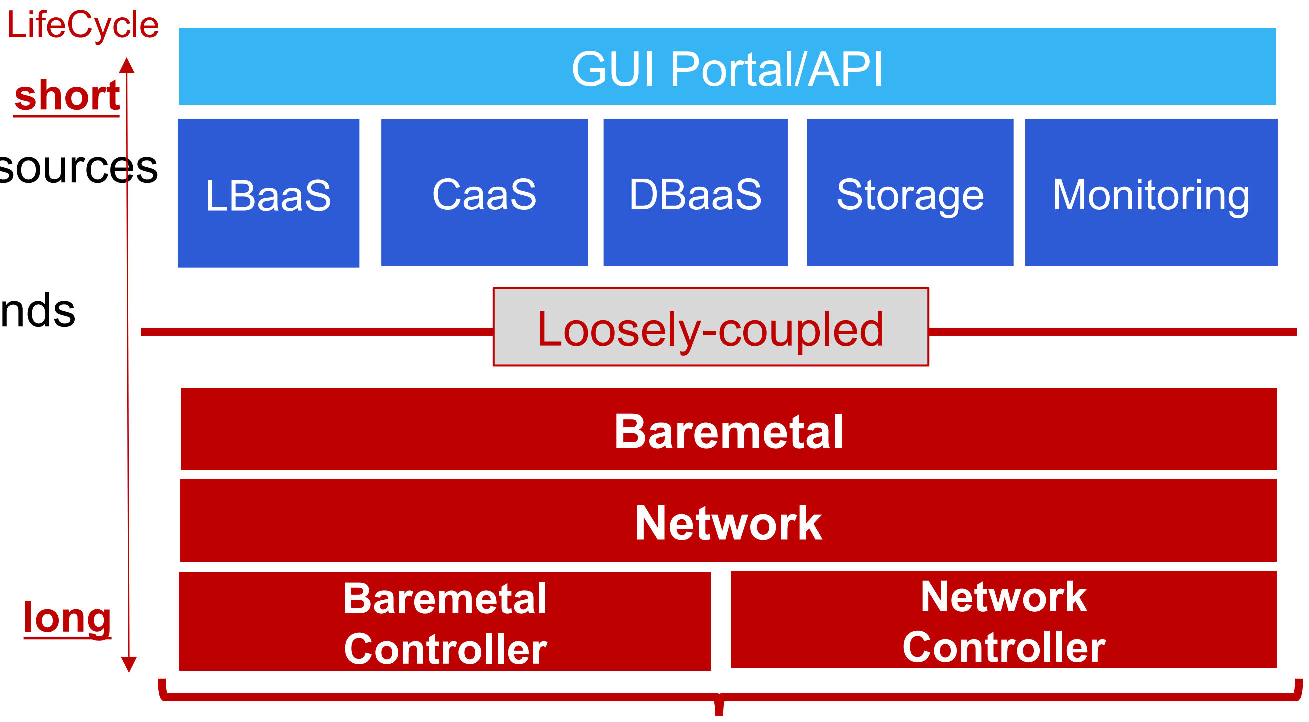
Advantages

- Provide stable and high-performance compute resources
  - No noisy-neighbor problem
- Simple and resistant to changes in technology trends
  - Offload high-level function as Managed-service
- Always up to date
  - Controllers are our permanent software assets

Trade-off

Require high infrastructure skills for bare-metal users

- Ensuring redundancy in case of bare-metal failure
- No physical resource abstration
- Resource utilization optimization should be covered by user product



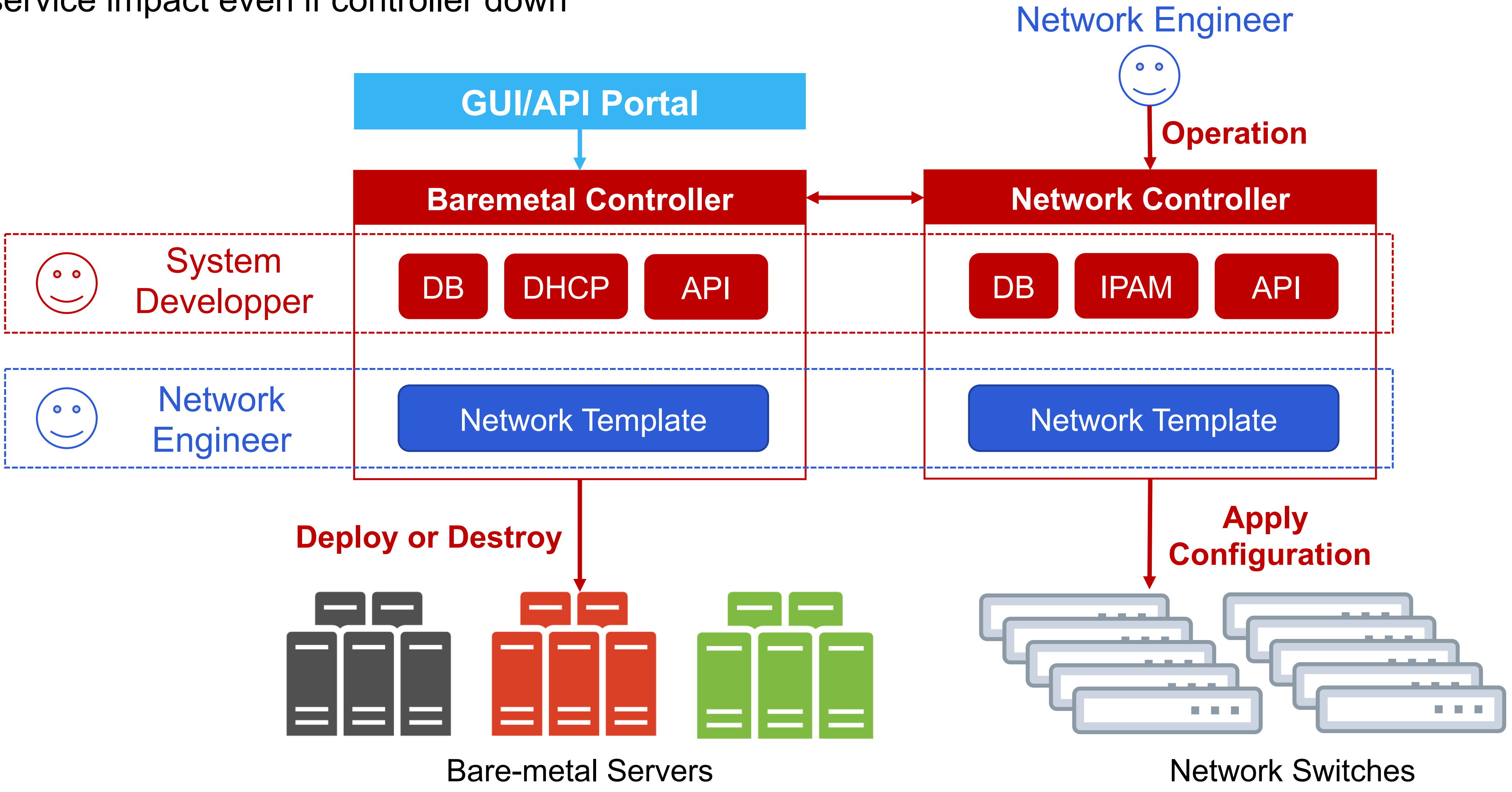
Sustainable Core-Infrastructure

- We have full control of the product life
- Apply improvement continuously
- Eliminate of the concept of upgrading

Develop new services using a combination of Managed-services

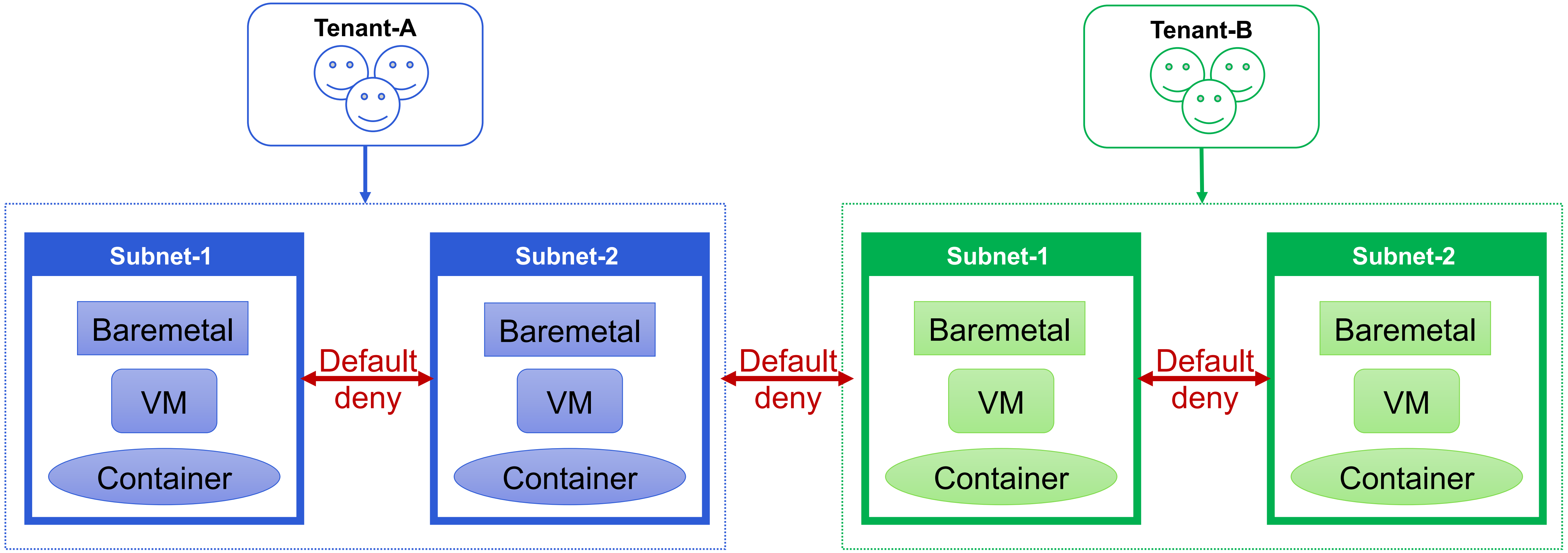
Baremetal / Network Controller

- Controllers are independent for each data center
- No service impact even if controller down



Multi tenant network design (Ideal) **Not adopted**

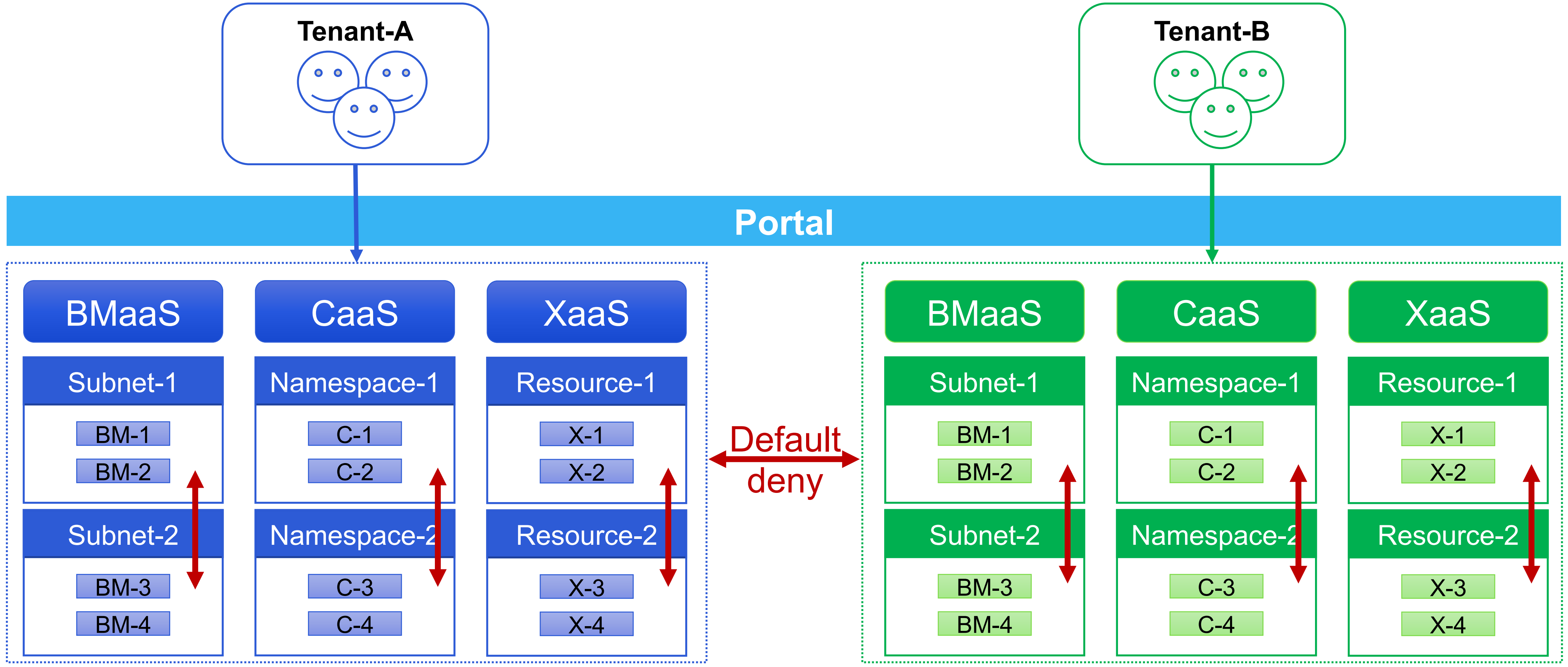
- Running different workloads in same subnet should be easier to manage?
  - Run Bare-metal, VM and Container in same subnet → Simple management?
- This approach was discarded because it's needed tightly coupled between Core-Infrastructure and Platform



Multi tenant network design **Adopted**

Separate tenant resources provided by each mechanism on the Managed-service

- Portal provides centralized management to handle different workload resources easier
- Example) Core-Infrastructure side provides isolation of bare-metal by subnets



Goal 2

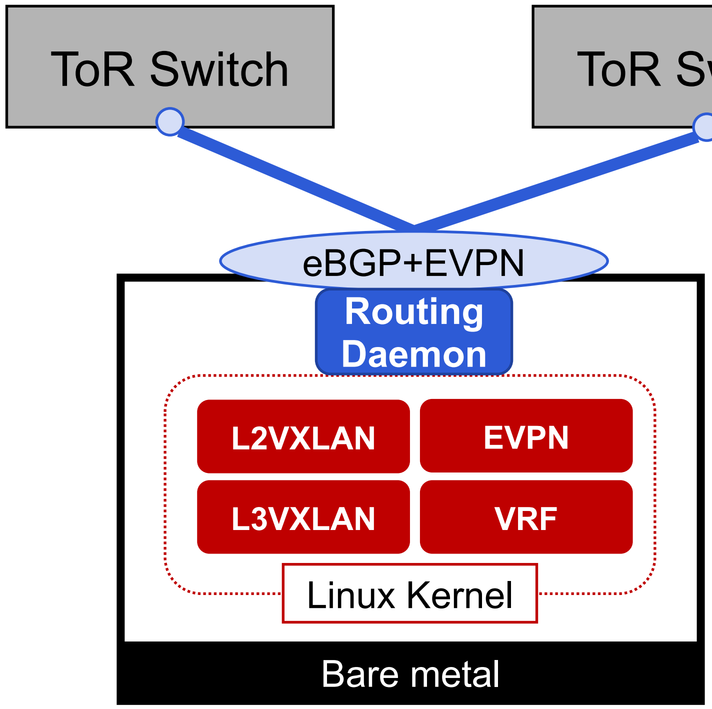
Stable + Scalability+ Easy-operation Network

How to realize L3 Bare-metal?

We aimed to eliminate MLAG from ToR layer to avoid a difficult situation

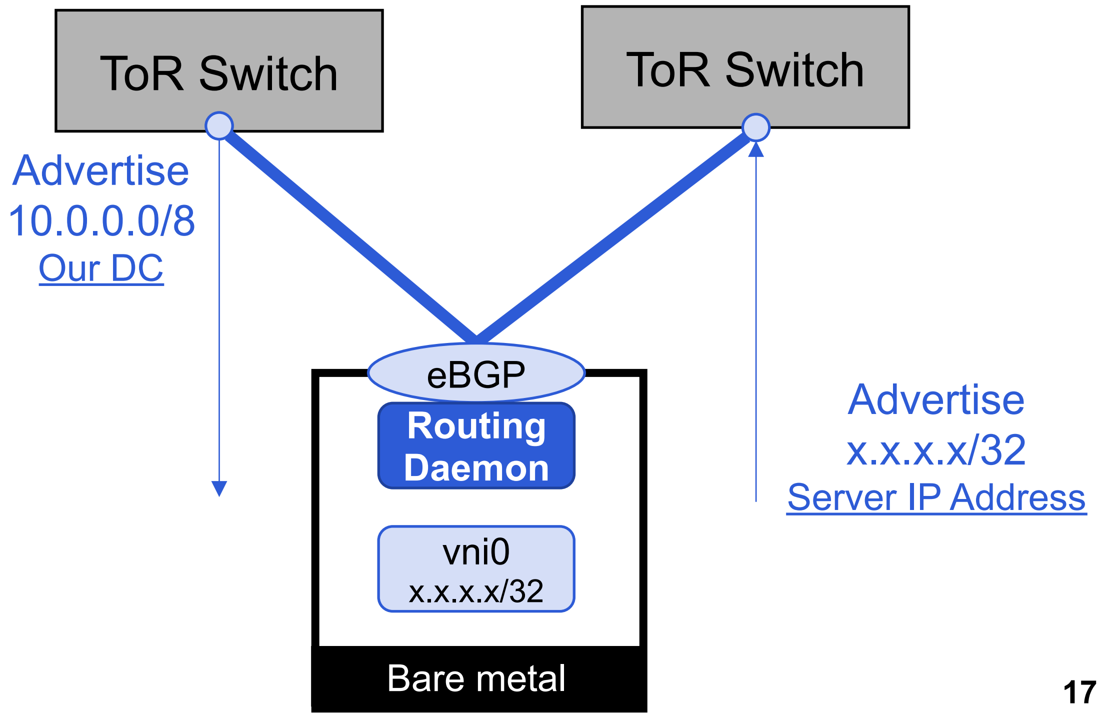
Approach 1: **VXLAN/EVPN on the Host** **Not adopted**

- (Good) Rack-wide L2 network can be deployed without MLAG
- (Bad) High OPEX
  - Complicated network configuration on the host
  - Difficult trouble shooting
  - Many features depend on Linux Kernel



Approach 2: **Routing on the Host** **Adopted**

- Switch only needs to support BGP
  - Reduce the probability of switch bug
  - Low learning cost and easy trouble shooting
  - More NOS selection due to simpler requirement
- Simple redundancy thanks to ECMP
  - Goodbye MLAG

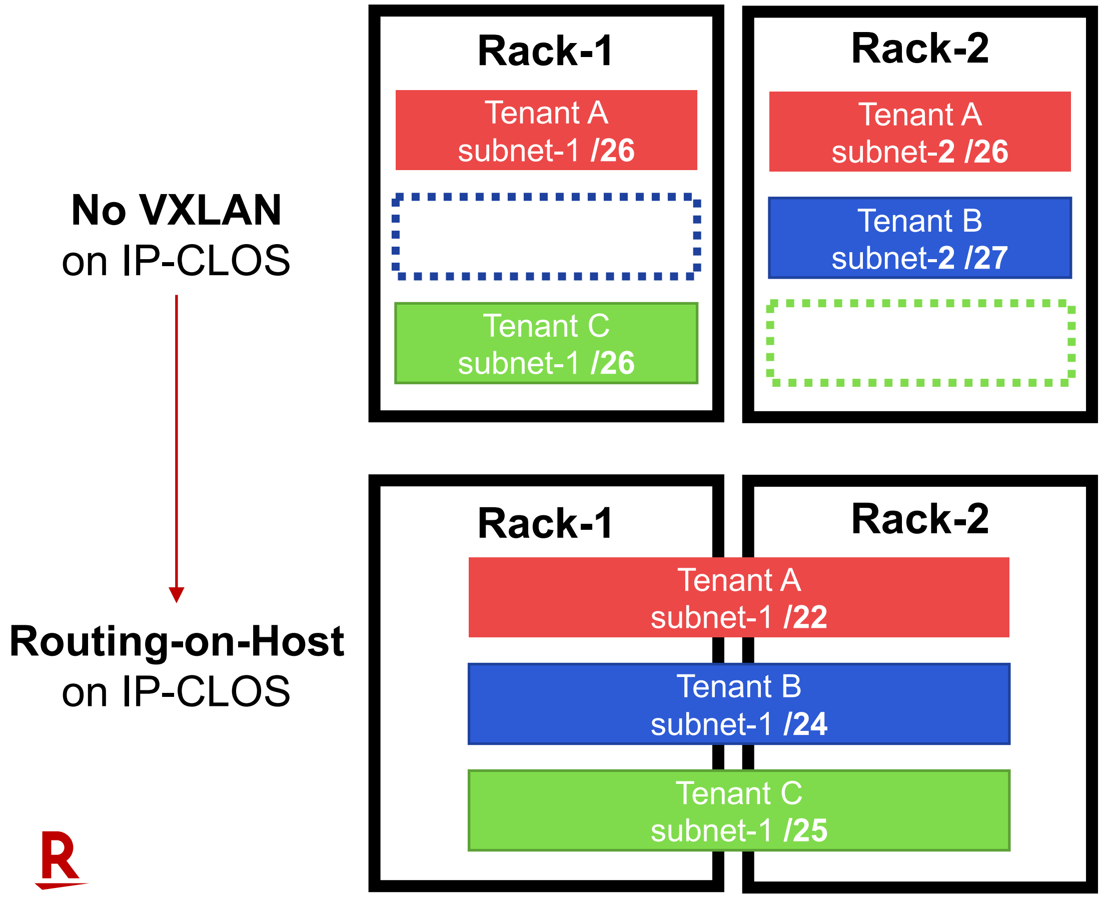


Usecase of L3 Bare-metal

Rack wide Mobility

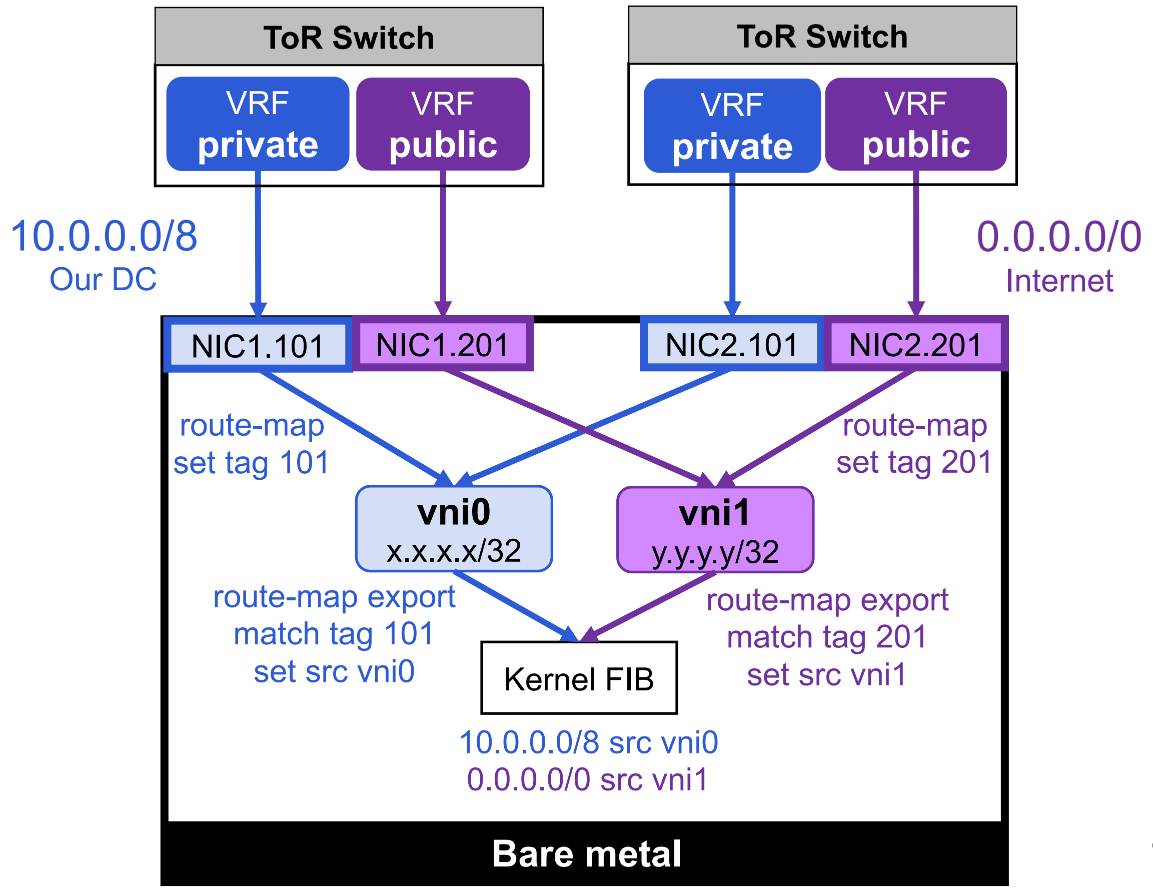
Subnets can be deployed across the racks

- Select a server regardless of rack location
- Simplify subnet management



Multi Network

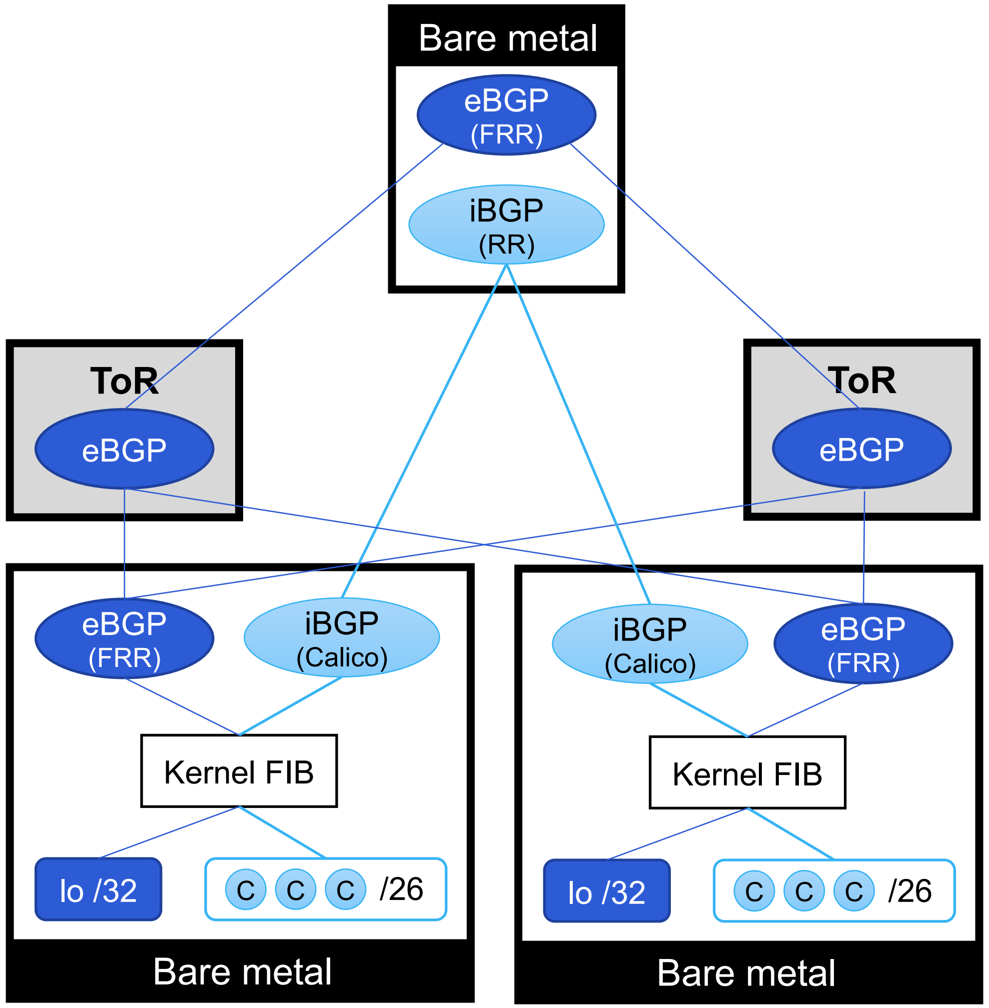
- Provide some VRF directly to the bare-metal
- Ex) LBaaS nodes have multiple virtual NICs
- Set route-map automatically as network catalog



Kubernetes Networking on L3 Bare-metal

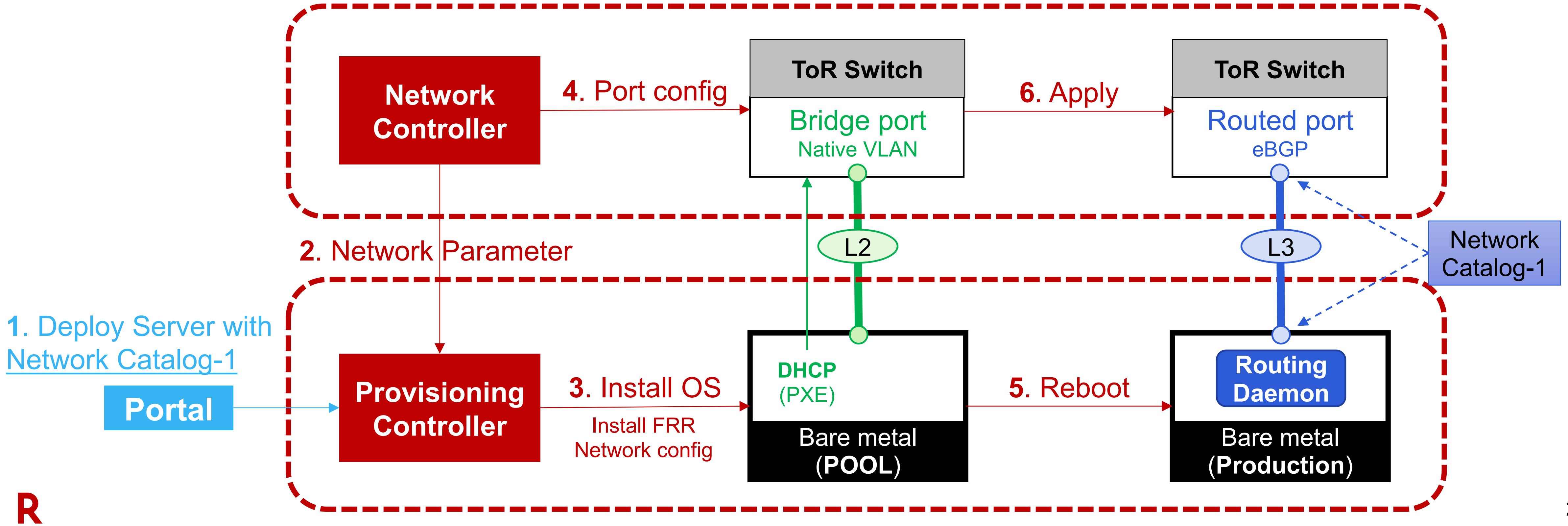
Consider the best network design for CaaS on Bare-metal

- Demarcation point of responsibility
  - FRR: Managed by Core-Infrastructure side
  - Calico (BIRD): Managed by CaaS
- No concern about explosion of container routes
  - Switch doesn’t learn the route of the container
  - PoC: Container-aware Load balancing by LBaaS
- Notice
  - Create a FRR container for Fedora-CoreOS
    - systemd-nspawn, not docker for user
  - Change BGP port number on FRR
    - BIRD:179, FRR:20179
  - Disable ECMP on CaaS Bare-metal
    - Use local-preference (Transmission: Active/Standby)
    - Why?: BIRD couldn’t process IPv4 link-local provided by FRR IPv6 unnumbered (Just my assumption)



Provisioning of L3 Bare-metal

- Providing a self-service bare-metal server for users
  - Select Linux distribution, subnet and network catalog (L2 or L3 etc..)
- ToR switch port configuration is changed dynamically
  - Deploy a server: Change switch port connection from L2 to L3
  - Back to pooled server: Change switch port connection from L3 to L2



Experience of L3 Bare-metal server

- Consider route hijacking
  - User has root privileges so can modify FRR configuration, it has risk of network failure
  - Set prefix-list on the switch downlink to filter invalid host route from the bare-metal

- Sharing knowledge for Bare-metal users
  - Example) Host network got down because the user disabled IPv6 (BGP was disconnected due to IPv6 unnumbered)
  - Example) Small hack is required to deploy Kubernetes via kubeadm (Set static route temporary)

- Nees to be adjusted for new Linux distribution
  - Fine-tune a new distribution image by Core-infrastructure side

- Doesn’t support L2 connectivity requirements
  - L2 prerequisite middle-ware doesn’t work (Redis+Sentinel, keepalived etc..)
    - Planning alternative network solution (Anycast IP etc..)
  - Providing L2 network catalog also as normal IP-CLOS
    - Subnet is closed by rack so subnet management is complicated

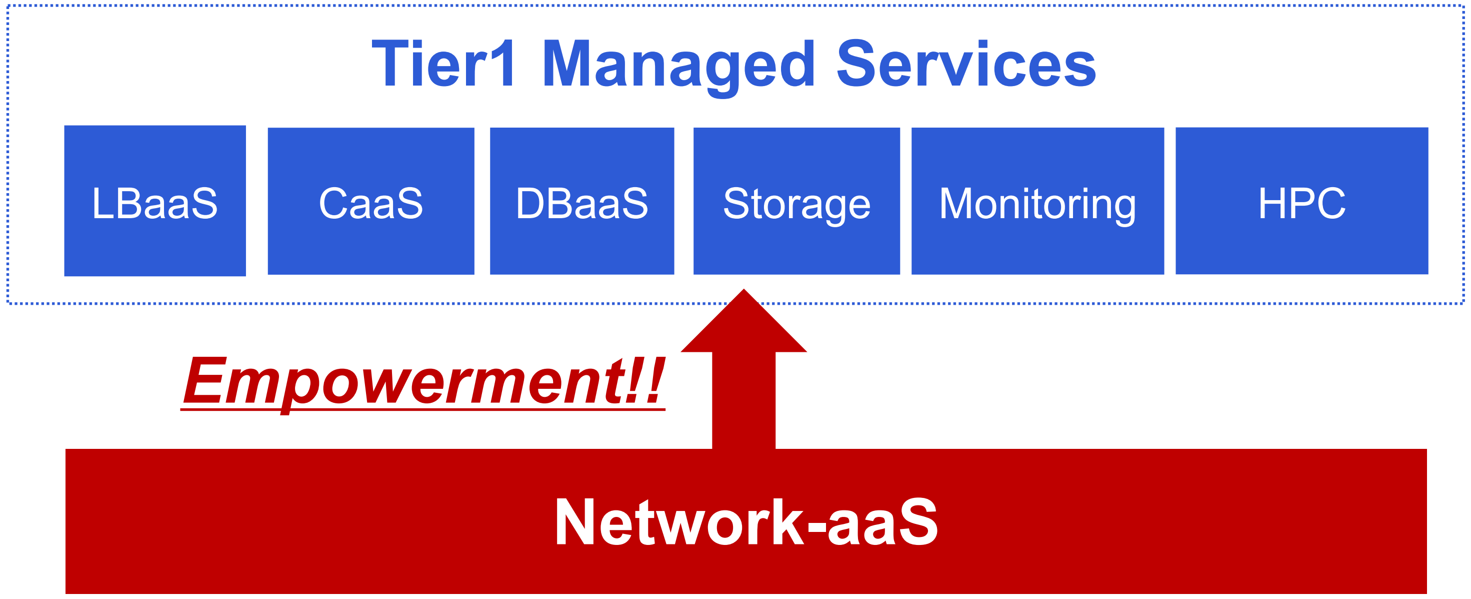
Goal 3

Network as a Service

Network as a Service

Providing network functions with built-in security mechanism

- **Distributed Firewall**
  - How to isolate a flat network as a multi-tenant network?
- **Internet Gateway**
  - Provide Internet connection without going through a proxy

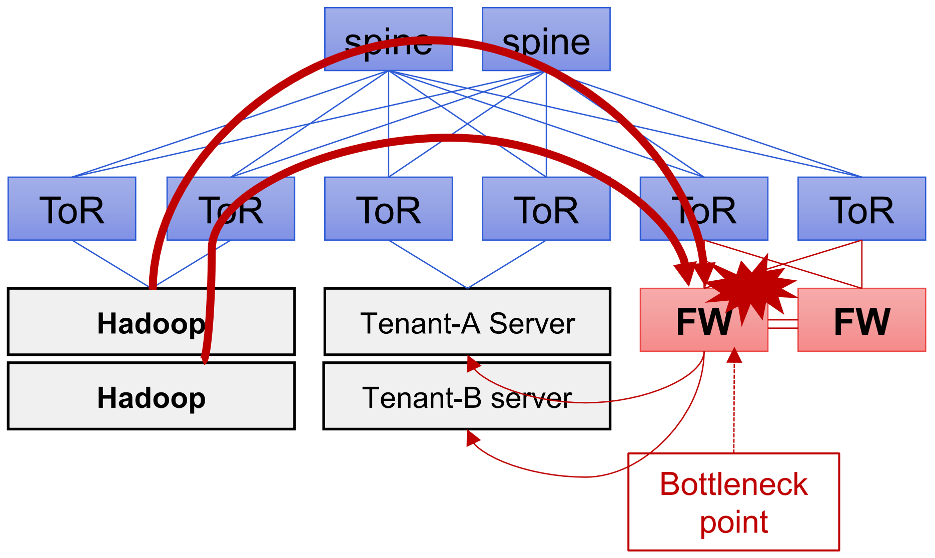


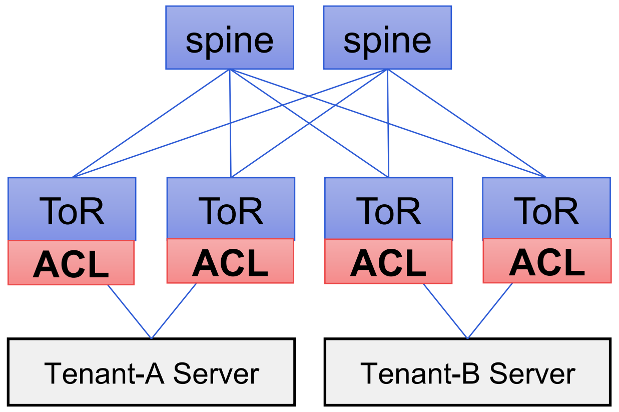
Multi subnet isolation

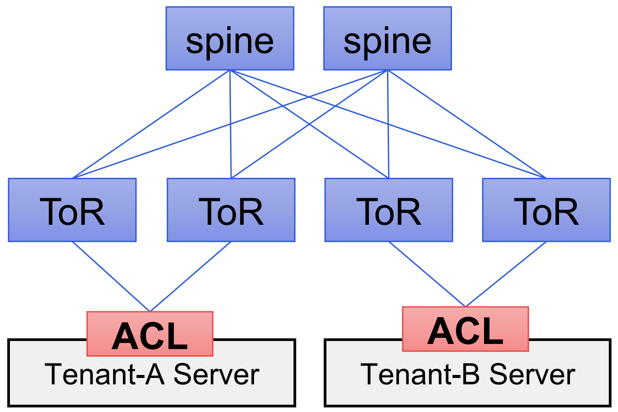
Requirement: Default deny between subnets but allow the traffic flexibly as needed

- Bare-metal doesn't have a useful function like the security group provided by OpenStack
- Flexible traffic control by using 5-tuple based on Host/Subnet units

Approach	1.Firewall Appliance	2.Switch based ACL	3.Distributed Firewall Adopted
CAPEX	Very expensive	No problem	Original Implementation
Limitation	No problem	Upper limit by ASIC	No problem
OPEX	Depends on the method	Difficult	Depends on the method



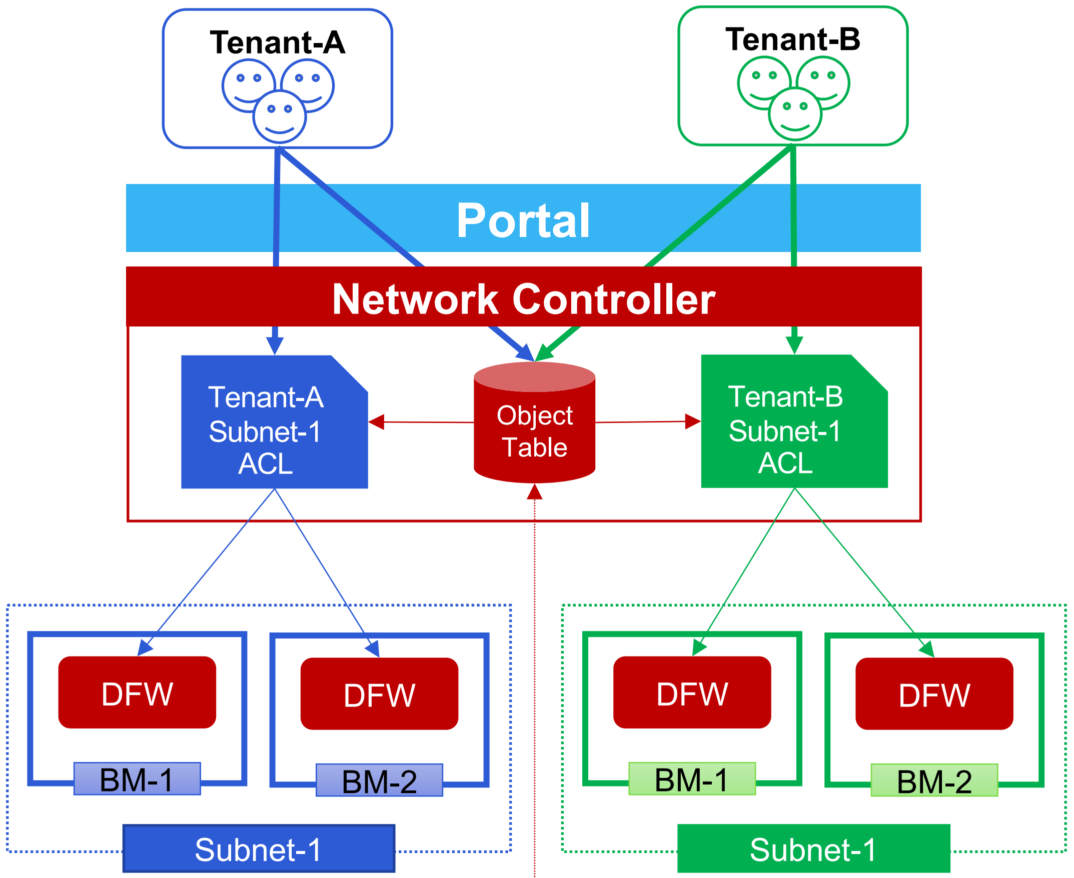




Distributed Firewall Pre-release

Protect bare-metal in own subnet from outside

- Deploy a firewall container on the user's bare-metal
  - Used systemd-nspawn like sidecar of k8s
  - Apply default-deny
  - DFW agent get the subnet ACL from network controller
- Helper function: Object Table
  - Source/Destination IPAddress is defined as object
  - User can select the objects when set ACL
- Challenge: Unable to enforce the rules
  - User has root privileges so DFW can be purged
  - Health check function to check agent status



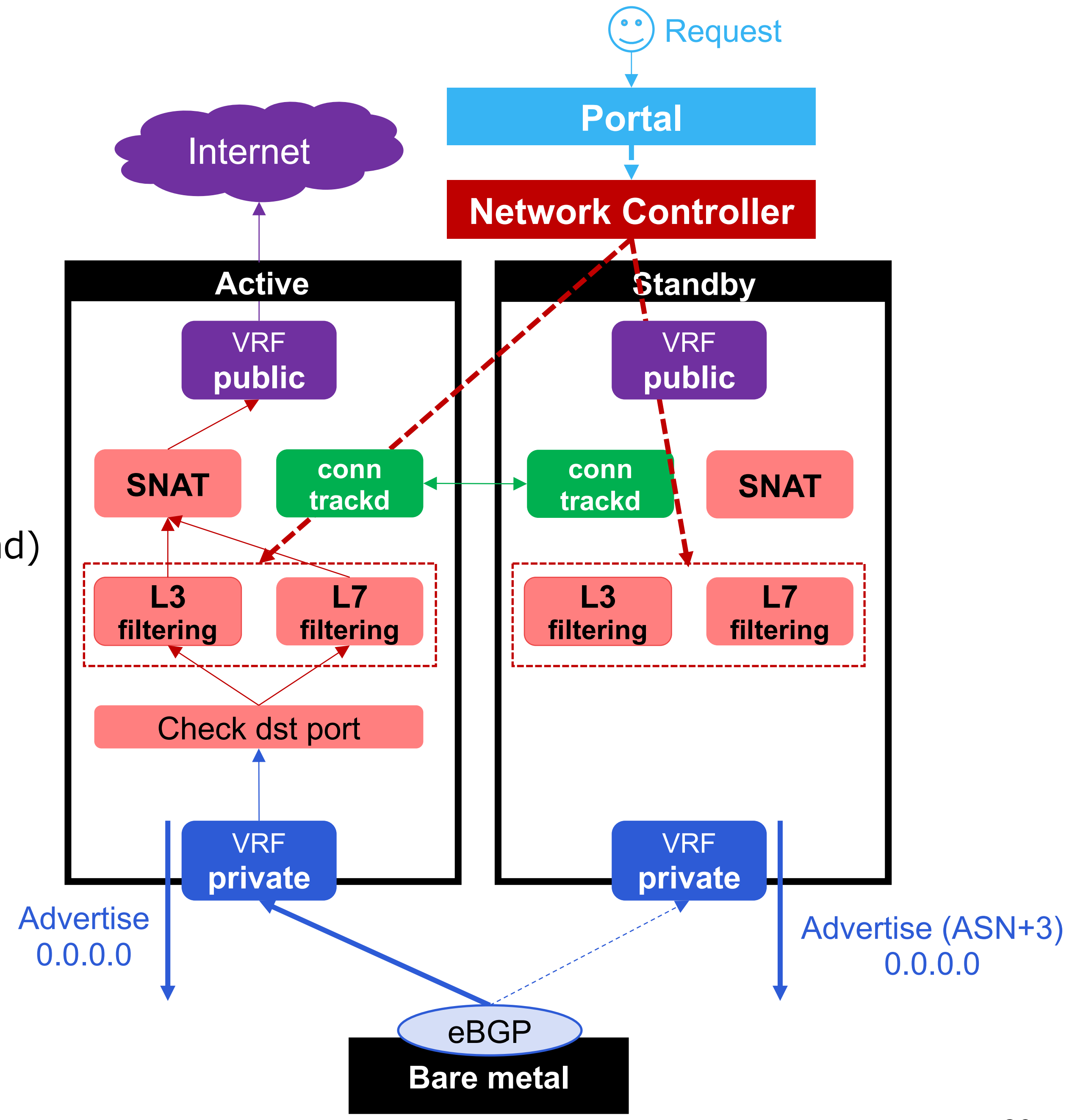
Object	IPAddress	Owner
Service-A	x.x.x.x./24	Tenant-A
Service-B	- y.y.y.y/24 - z.z.z.z/25	Tenant-B

Internet Gateway

PoC

Provide Internet connection with under control

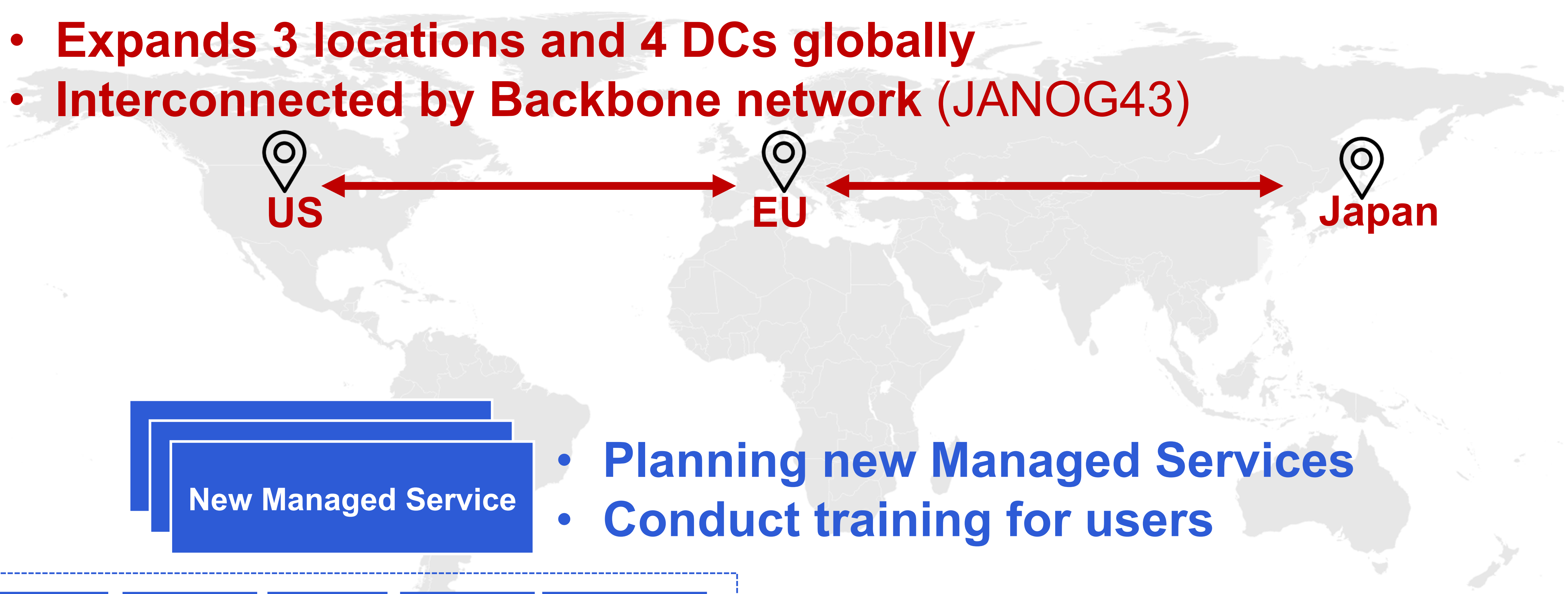
- Advertise default-route for all bare-metal
- Active - Standby design
  - Constructed by 2 bare-metal
  - Synchronizatoin of session table (conntrackd)
  - Controlled by arbitration script (ASN path-prepend)
- Filtering
  - L7: URI filtering (Squid)
  - L3: Protocols other than HTTP/HTTPS (nftables)
- Challenge
  - Active – Active design
  - Tune-up various parameters



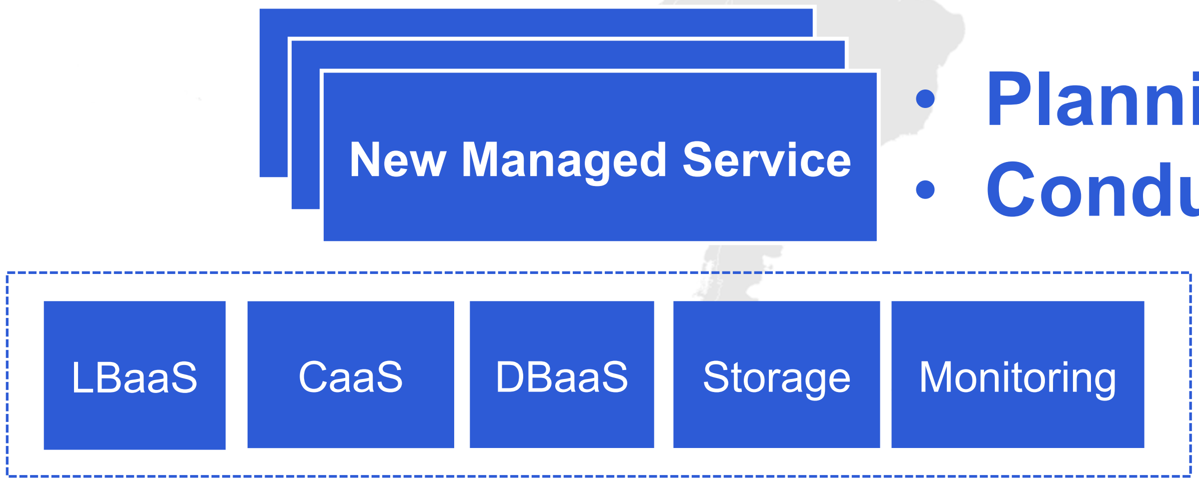
Current Status

Current Situation

- **Expands 3 locations and 4 DCs globally**
- **Interconnected by Backbone network** (JANOG43)



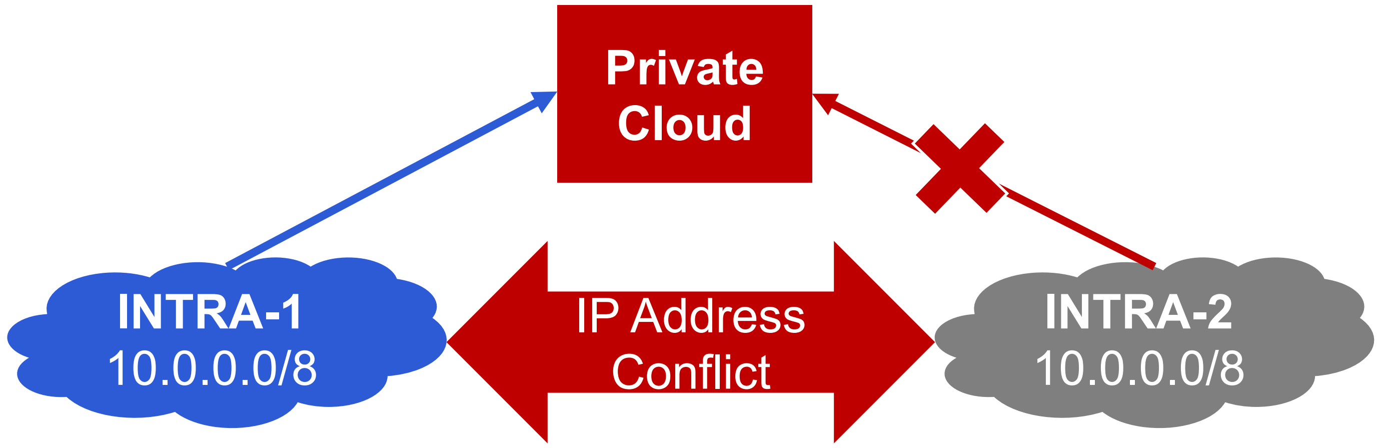
- **Planning new Managed Services**
- **Conduct training for users**



https://power-point-design.com/ppt-design/world-map-for-powerpoint/

Known Issues

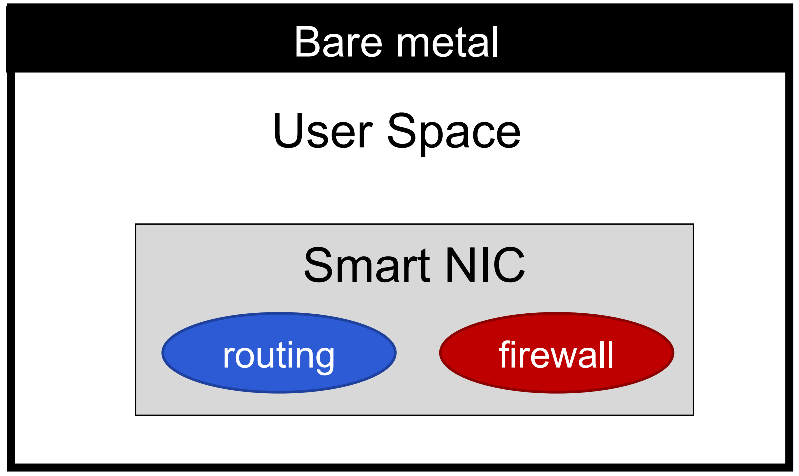
- Private IPv4 address shortage near the future
  - Mass IP address space is consumed when building a new POD
  - Since it was designed as flat network without overlay
  - Start to consider using IPv6 for internal network
- Network security policy when connecting to the existing platform
  - Some existing platforms are not designed as multi-tenant
- Overlapping IP address space with existing network
  - Create a dedicated VRF and NAT (PoC)



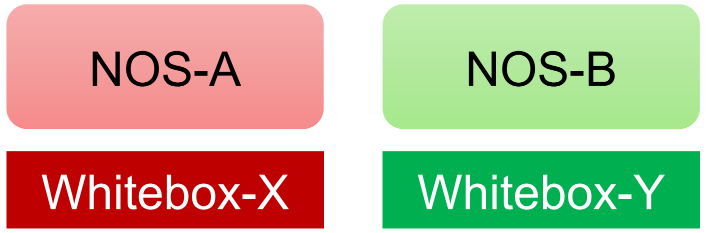
Next Challenge

Offloading network functions

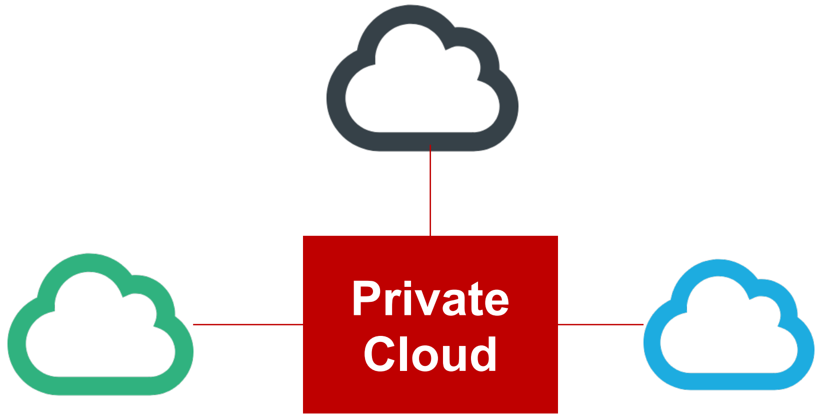
- We want to provide pure compute resource for users
    - Offload packet processing on Smart NIC
- User-inviolable control point in bare-metal
    - Run 1VM on hypervisor etc..



Next Whitebox NOS



Enhanced connectivity with Public-Cloud



Summary

<u>Launched new infrastructure which can respond to the rapidly changing world</u>

Built a private cloud to accelerate our business

- Managed service oriented, Simplified Core-Infrastructure based on Bare-metal
- Support multi-tenancy + Stable and scalable network by Routing-on-Host
- Provide useful network functions as Network-aaS to enhance the productivity of users

Discussion

Rakuten