



KubeCon



CloudNativeCon

Europe 2026

#KubeCon #CloudNativeCon

# Evolving Baremetal-as-a-Service: Secure Multi-Cluster Networking and Service Identity Automation

**Yushiro Furukawa & Mitsuhiro Tanino**  
**LY Corporation**



# Who are we?



KubeCon



CloudNativeCon

Europe 2026



## Yushiro Furukawa

Senior Software Engineer in LY Corporation

- Contributed OpenStack neutron, firewall-as-a-service, Ironi for several years
- Contributed metal3-io with network switch configuration for a few years
- Product Owner of Flava Baremetal
- Like Rubic-cubes



# Who are we?



KubeCon



CloudNativeCon

Europe 2026



## Mitsuhiro Tanino

Senior Software Engineer in LY Corporation

- Contributed to OpenStack Cinder for several years
- Contributed to Kubernetes sig-storage for several years
- Operates hyper-scale OpenStack cloud environments
- Develops features to enhance private cloud stability

- 1. Introduction of Our Private Cloud and Baremetal-as-a-Service**
- 2. Dive into Baremetal-as-a-Service**
  - **Unified API**
  - **C-Plane — Built Entirely on Kubernetes**
  - **Integrated Networking&Security**
- 3. Wrap-up**

# Introduction of Our Private Cloud

- LY Corporation was formed through the merger of Z Holdings, which includes Yahoo Japan Corporation, and LINE Corporation
- Flava is our next-generation private cloud, designed to provide a unified, **scalable**, and **reliable** infrastructure for all our services
- It is built on a massive OpenStack on Kubernetes, providing a wide range of services from IaaS to PaaS for our developers



**kubernetes**

# Multi-Region & AZ Topology — Full Picture



KubeCon



CloudNativeCon

Europe 2026

## Three levels of physical hierarchy:

- **Rack / Hypervisor** → Physical racks and baremetal nodes — the smallest unit of management
- **Availability Zone (Building)** → Physical buildings or independent power domains
- **Region (Data Center)** → Geographically separated Data Centers

## Multiple regions, mirrored for Disaster Recovery:

- Each Region runs identical configurations
- If an entire region becomes unavailable, operations continue from the other region
- Within each region, multiple AZs provide building-level fault isolation

## Why this topology matters for our architecture:

- AZ failure (building-level) must not take down the management plane
- Region failure (DC-level) must be recoverable — Disaster Recovery posture
- We define “Failure domain” as a rack



# Scale of Flava (as of 2025-06)



KubeCon



CloudNativeCon

Europe 2026



|                    |              |
|--------------------|--------------|
| Hypervisors        | 500+         |
| Virtual Machines   | 9,200+       |
| Baremetal Servers  | 3,000+       |
| Regions            | 2+           |
| Availability Zones | 6 (3AZ x 2)+ |

# Flava

# Why Baremetal-as-a-Service?

## AI / ML

Direct GPU and hardware access for maximum training and inference throughput

## Legacy system migration

Applications with specific hardware requirements or dependencies  
Systems requiring dedicated hardware for compliance reasons

## Storage workloads

Supports flexible storage configurations, such as high-density or high-performance storage  
Enables direct device access and prevents redundant replication at the software layer

## Software with CPU core-based license

Allows use of lower-core CPUs to minimize license fees



KubeCon



CloudNativeCon

Europe 2026

# Dive into Baremetal-as-a-Service

# Three Pillars of our Baremetal-as-a-Service



CloudNativeCon  
Europe 2026

## 1. Unified API

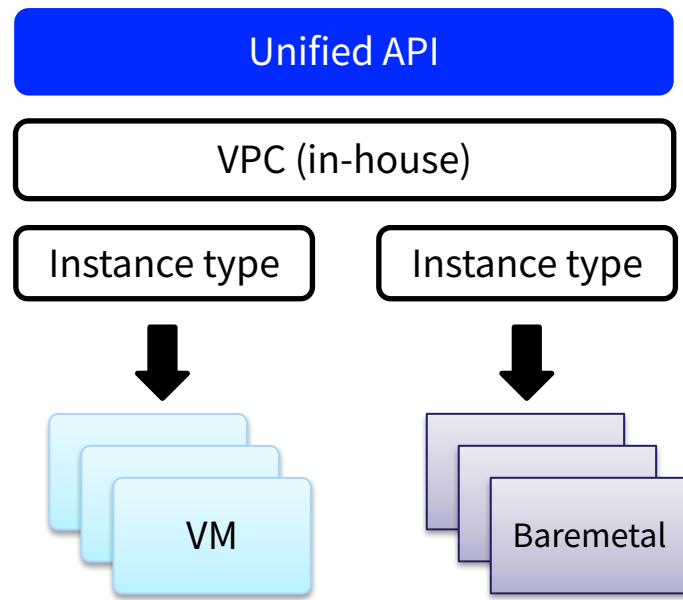
**One API endpoint** for both VMs and Baremetal Servers

## 2. C-Plane — Built entirely on Kubernetes

All control plane components run on Kubernetes → highly available across multiple AZs

## 3. Integrated Networking&Security

Apply network policies by VPC and X509 certificate to Baremetal Servers automatically at provisioning



# Unified API — Same Interface, Different Backend



KubeCon



CloudNativeCon

Europe 2026

From the user's perspective: VM and Baremetal are identical.

```
# Provisioning a VM
$ openstack server create --flavor 8vCPU_32GB_120GB --image ubuntu-img --network vpc-net my-vm-server

# Provisioning a Baremetal server
$ openstack server create --flavor 52pCPU_256GB_S960GB.metal --image ubuntu-img.metal --network vpc-net my-baremetal-server
```

- Both VMs and baremetal are provisioned through OpenStack Nova.
- But the user never needs to think about what's running underneath.
- **Under the hood: fundamentally different.**

|               | VM                   | Baremetal                  |
|---------------|----------------------|----------------------------|
| Runs on       | Hypervisor           | Directly on hardware       |
| Network       | Virtualized          | Physical network           |
| Driver        | Standard Nova driver | Custom baremetal driver    |
| Kernel        | Shared               | Dedicated                  |
| Boot          | Virtual disk         | IPMI PXE boot + OS install |
| Power control | Hypervisor API       | IPMI                       |



KubeCon



CloudNativeCon

Europe 2026

# C-Plane — Built entirely on Kubernetes

# Baremetal-as-a-Service— C-plane on Kubernetes



KubeCon



CloudNativeCon

Europe 2026

The entire control plane is Kubernetes-native.

## C-Plane (K8s Cluster): runs entirely on Kubernetes

- All management components run on Kubernetes — self-healing, declarative management, multi-AZ resilient



### K8s Cluster (C-Plane)

OpenStack

IPMI Management

OS Installer

Custom Controllers

## D-Plane (Physical Servers): orchestrated by C-Plane

- Not on Kubernetes — but fully controlled and orchestrated by the C-Plane

### External Systems: integrated via API

- Automatically updated when a baremetal instance is created, so users can access their server immediately

#### Baremetal (D-Plane)

Physical Nodes

PM Management Agent

VPC Agent

#### External Systems

Asset DB

DNS

# Multi-Region & AZ Topology C-Plane Detail



KubeCon



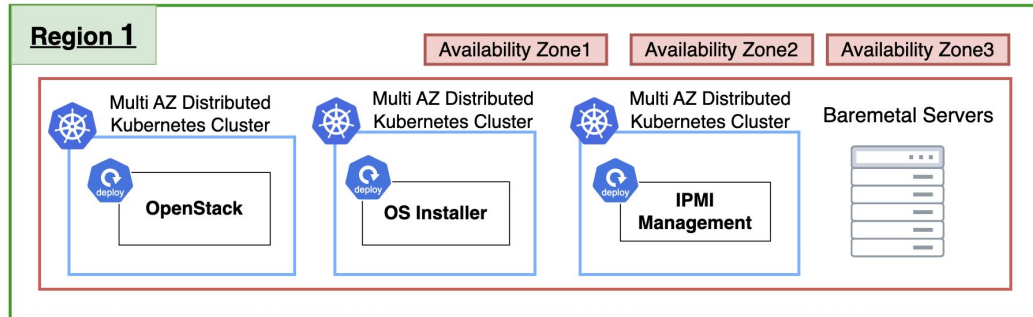
CloudNativeCon

Europe 2026

The management plane is itself highly available.

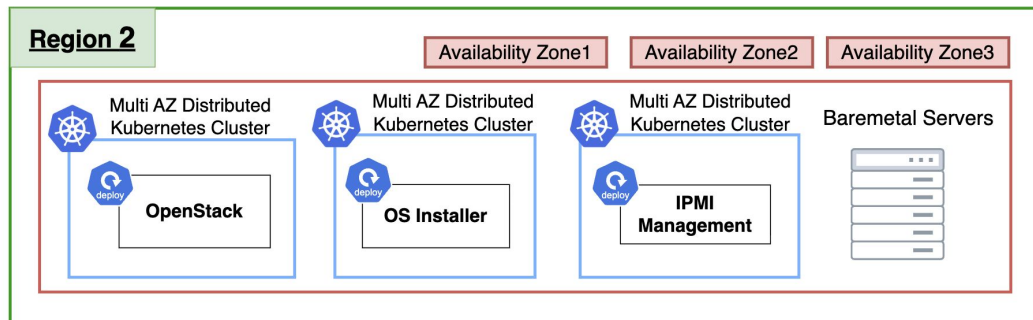
## Multi-AZ Distributed K8s Cluster

- The management K8s cluster is distributed across all three AZs within a region
- Even if one AZ (building) fails, the management plane continues operating
- Baremetal servers in surviving AZs remain fully manageable



## Multi-Region Deployment

- The same C-Plane configuration is deployed in both regions
- Ensures DR policy — regional disaster does not mean loss of management capability



# C-Plane Deep Dive — IPMI Management



KubeCon



CloudNativeCon

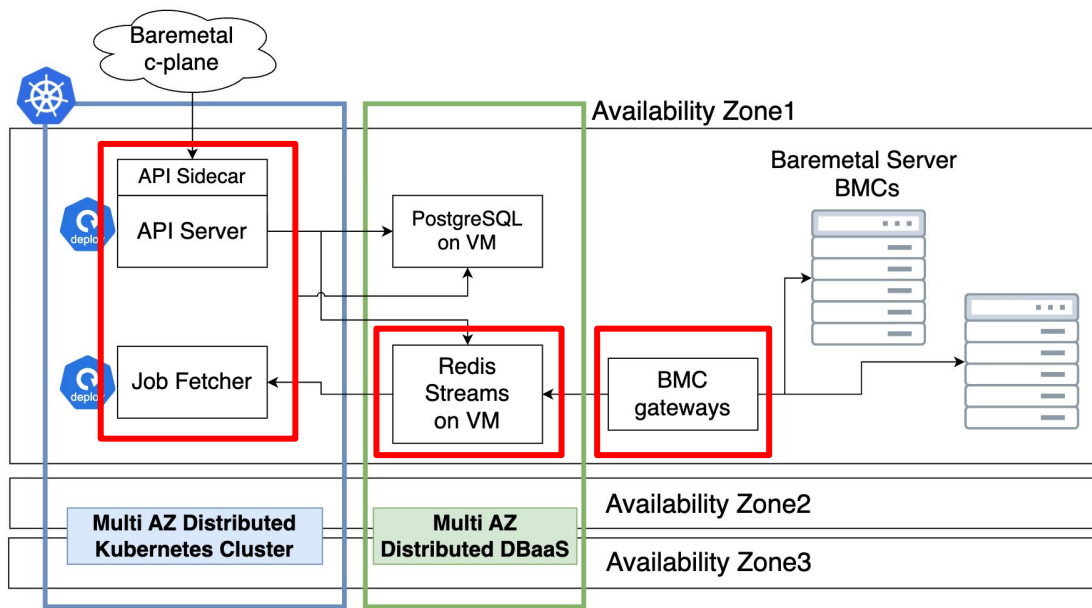
Europe 2026

**Hardware control is slow and unreliable. Kubernetes helped us solve that.**

## The challenge with IPMI:

- IPMI operations (power on/off, reboot) can take 30 seconds or more
- Hardware BMCs are sometimes unresponsive or return inconsistent results
- At our scale, hundreds of IPMI operations may run at the same time

-  **API Server** — responds immediately, never blocks with k8s self-healing
- **Redis Streams** — decouples fast API from slow hardware (*VM-based DBaaS*)
- **BMC Gateways** — executes IPMI, deployed per datacenter room (*physical server*)
-  **Job Fetcher** — collects results asynchronously



# C-Plane Deep Dive - OS Installer



KubeCon



CloudNativeCon

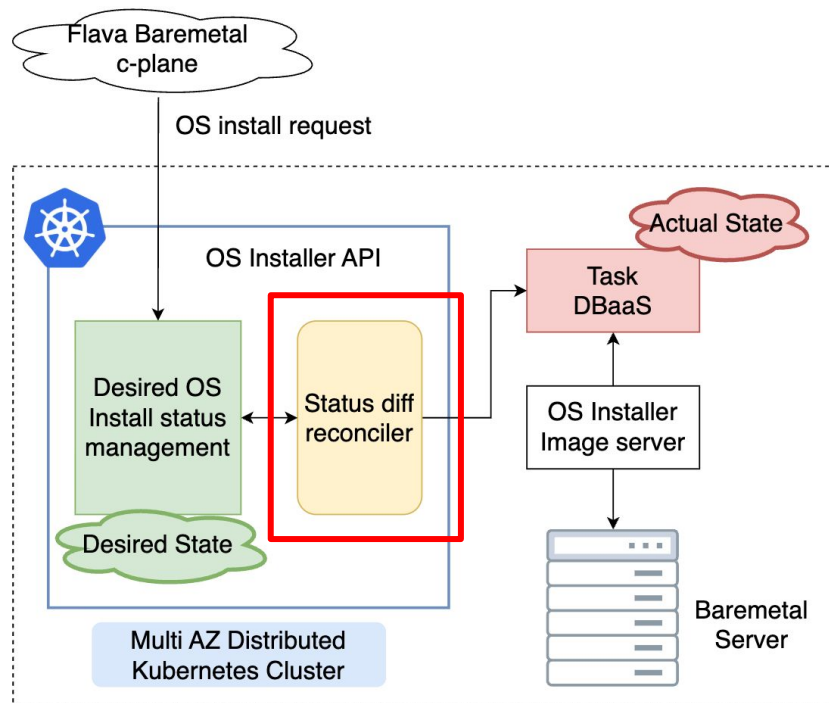
Europe 2026

OS provisioning follows the same Kubernetes patterns as everything else.

How the reconciler mechanism works:

- **OS Installer API** — continuously watches desired OS installation state
- **Reconciler** — detects gaps between Task DB and actual server state
- **On failure** — reconciler automatically detects and retries

These k8s patterns abstracts complex physical OS deployment into standard, resilient API calls.



# Previous KubeCon Japan Recap — The Scalability Problem



CloudNativeCon

Europe 2026

## Moving Deployment procedure from Ansible to Kubernetes Custom Controllers brought dramatic improvement

At 6,000+ baremetal server compute manager deployment, our Ansible-based architecture could not keep up:

**Deployment time**



### Ansible-based

6,000+ Baremetal compute manager deployment takes more than 2 hours



**Finishes in few minutes**

**Resource In-efficiency:**

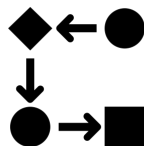


6,000+ pods requires 55+ workers (28vCPU 56GB Mem)



**10+ workers are enough**

**Operational Pain:**



Increases operational complexity and cost for IaaS Operator

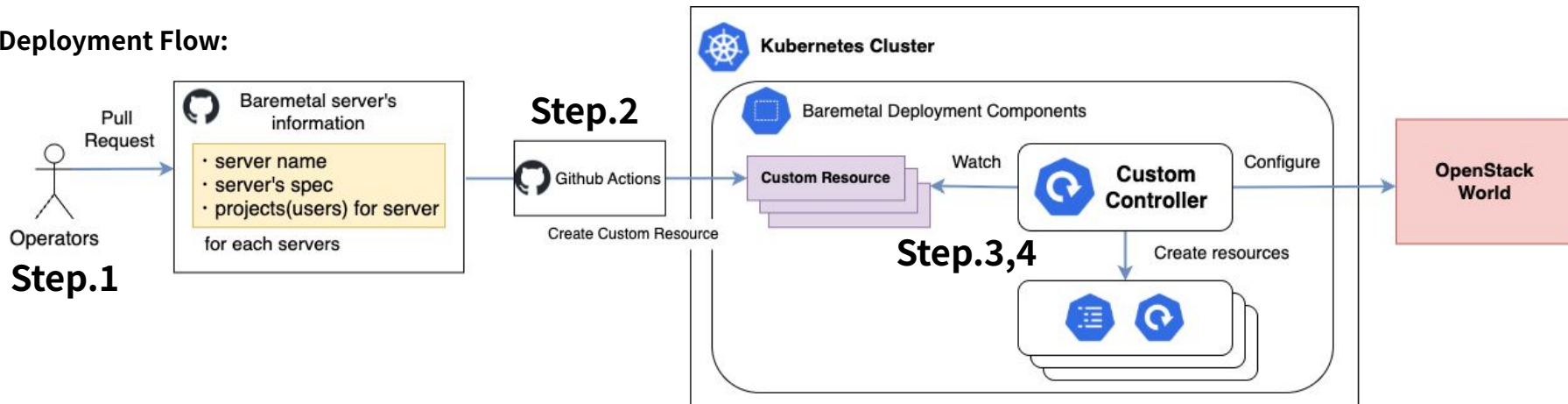


**With self-healing of custom controller, operator cost is significantly reduced**

# Previous KubeCon Recap — Custom Controller Solution

The solution: model every baremetal server compute manager as a Kubernetes Custom Resource

## Deployment Flow:



1. Baremetal operator registers baremetal server's information to git.
2. Merging PR triggers Github Actions then, Custom Resource manifests are created and applied automatically.
3. Custom Controller keeps watching new creation or change of custom resources
4. Custom Controller performs the deployment process like creating Kubernetes resources, configuring OpenStack settings, etc for each server

## Physical racks are modeled directly as Kubernetes Custom Resources.

- One CRD per rack
  - rack maps to a physical failure domain
- RackA has multiple server managers
  - Each server entry declares hostname, **hardware spec** (*openstackFlavor*), and **tenant** (*openstackProject*)
- All server manager configuration lives in Git
  - no manual registration needed

### Custom Resource Definition

spec:

```
rack: rackA
computes:
- hostname: baremetal-server-1
  openstackFlavor: small.metal
  openstackProject: test-project1
- hostname: baremetal-server-2
  openstackFlavor: medium.metal
  openstackProject: test-project2
- hostname: xxx
```

Config for one server

Servers in the same rackA

# Previous KubeCon Recap — Reconciliation Loop



KubeCon



CloudNativeCon

Europe 2026

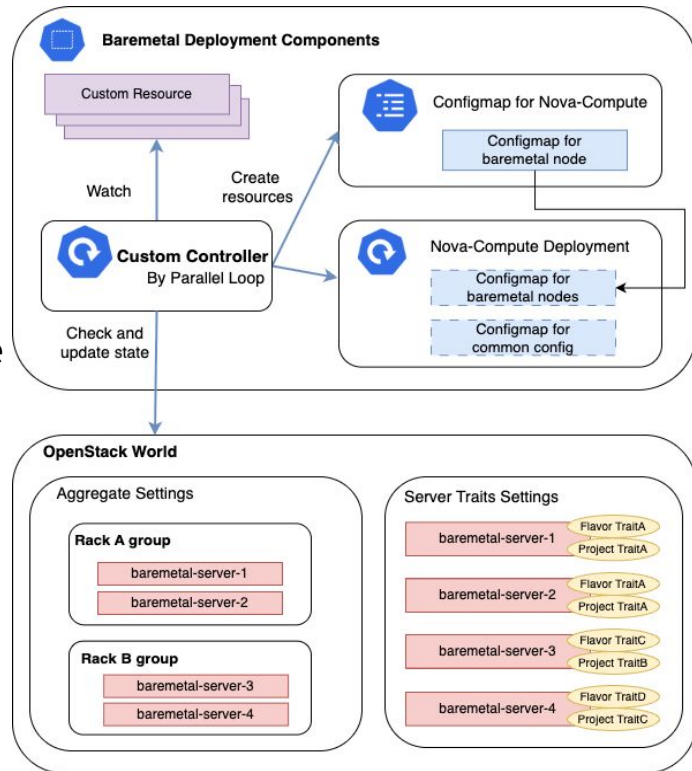
## What the controller does, and what we gained.

### Reconciliation loop actions per rack:

- Creates one ConfigMap per server (compute manager configuration)
- Creates one Deployment per rack — containing multiple compute manager containers, one per server in that rack
- Configures OpenStack required settings

### What Kubernetes gave us for free:

- **Parallelization** — multiple reconciliation loops run concurrently
- **Reduced worker footprint** — far fewer worker nodes required
- **Incremental deployment** — only changed resources are reconciled





KubeCon



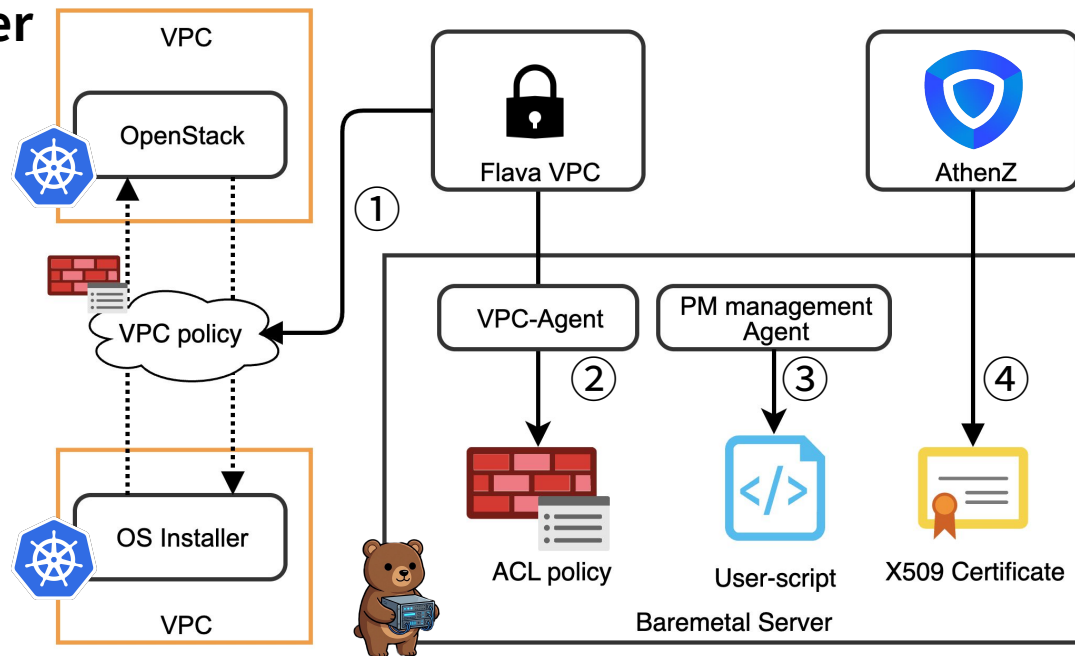
CloudNativeCon

Europe 2026

# Integrated Networking & Security

# Integrations of Network&Security

- ① ACL setting Between K8s clusters
- ② ACL setting For baremetal server
- ③ Run user-script
- ④ AthenZ integration



# ACL Setting Between k8s Clusters



KubeCon



CloudNativeCon

Europe 2026

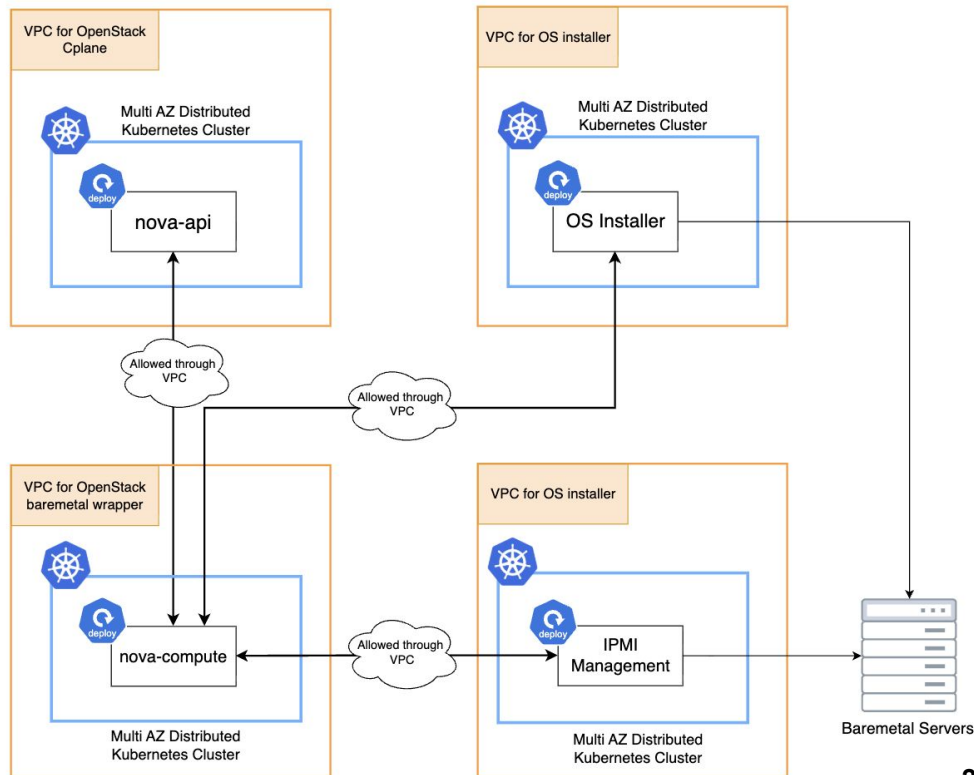
## Secure Service Integration

Management components (nova-api, nova-compute, OS Installer and IPMI management) are isolated in dedicated VPCs.

## Zero-Trust Pod-to-Pod Connectivity:

Automatically secured via VPC peering and dynamic ACLs.

Decoupled architecture ensures strict security boundaries between logical control and physical execution.



# ACL Setting for Baremetal Server



KubeCon



CloudNativeCon

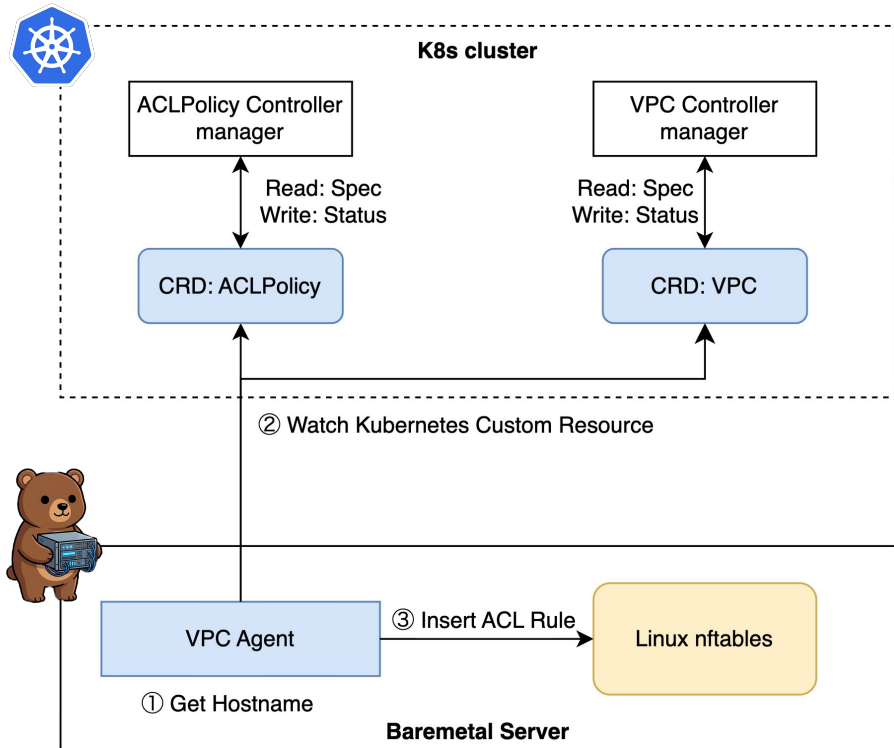
Europe 2026

## Kubernetes Cluster Side

- ACL policy and VPC Controller are managed as controller-manager and its CRD.
- They provide a desired ACL policies.

## Baremetal Server Side

- VPC agent keeps watching the desired status and inserts ACL rule into *nftables*



# ACL Setting for Baremetal Server



KubeCon

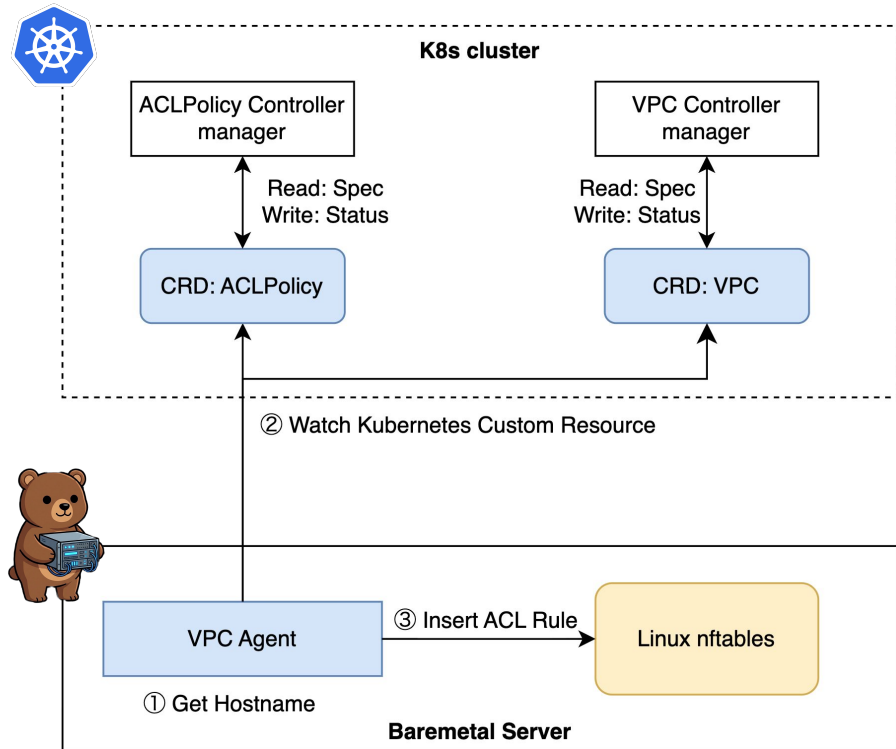


CloudNativeCon

Europe 2026

## Custom Resource Definition:

```
kind: ACLPolicy
spec:
  ingress:
    rules:
      - name: permit-from-x-to-y-443
        action: permit
        priority: 1100
        proto: tcp
        srcs:
          - prefix: 10.X.X.0/24
        sports:
          - max: 65535
            min: 1
        dsts:
          - prefix: 10.Y.Y.45/32
        dports:
          - max: 443
            min: 443
```



# Get and run user-script



KubeCon

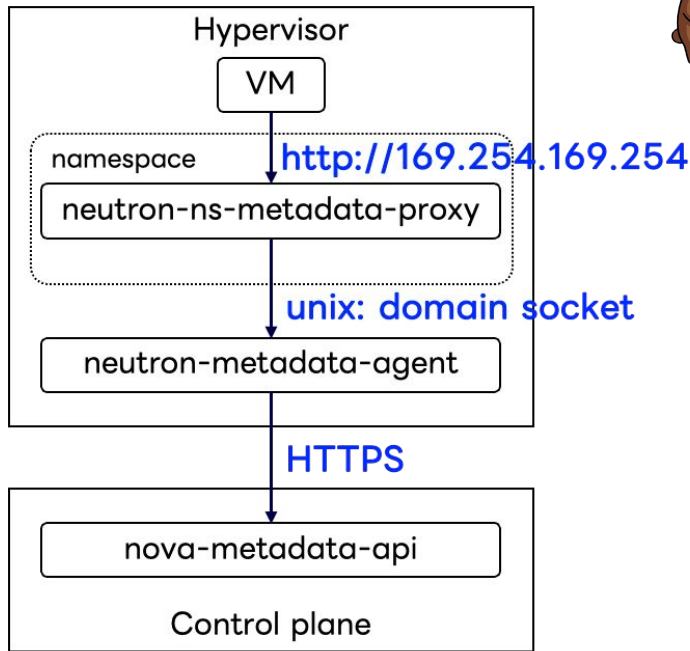


CloudNativeCon

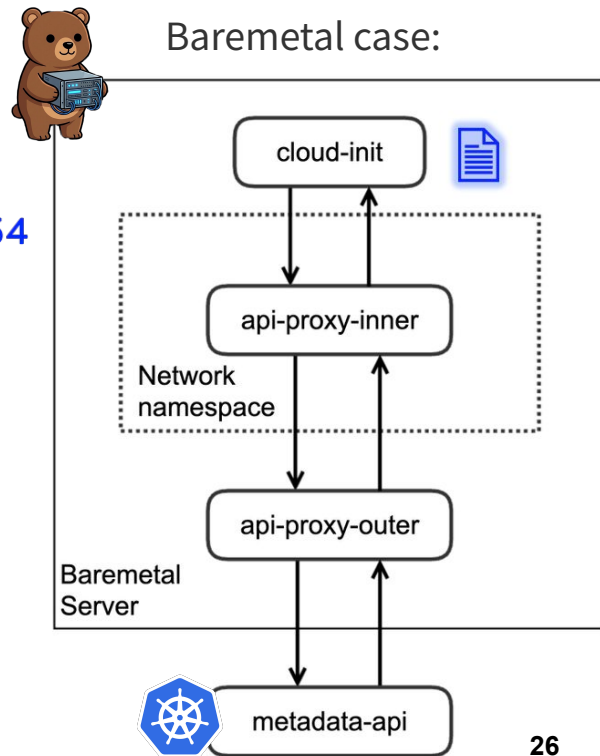
Europe 2026

- A lightweight agent, **PM Management Agent**, is installed in the base OS image and includes the following processes:
  - **api-proxy-inner**
  - **api-proxy-outer**
- This is totally like VM's architecture, and we made these processes by nginx.

Virtual Machine case:



Baremetal case:



# AthenZ Integration with Baremetal Server



KubeCon

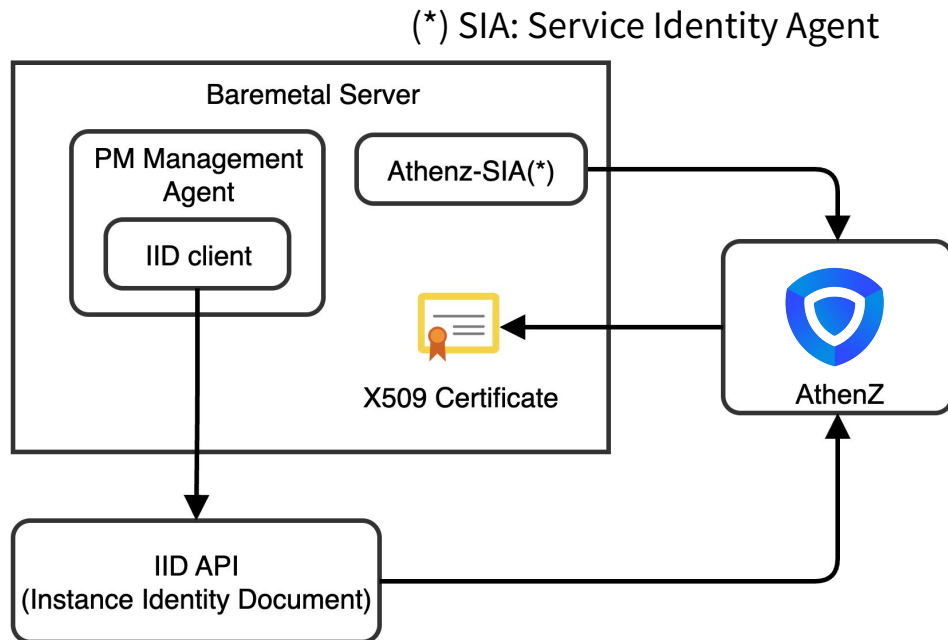


CloudNativeCon

Europe 2026

## Provides X509 certificate to Baremetal Server in booting process

- PM Management Agent calls IID (Instance Identity Document) API to issue IID
  - a. Get signed IID to verify instance identity
- Request X.509 Certificate from AthenZ-SIA
  - a. AthenZ verified the signed IID
  - b. AthenZ provisions X.509 Certificate to the baremetal server



## Three things that made Flava BMaaS cloud-native:

① **Nova + Kubernetes Custom Controllers** Declarative, scalable, self-healing baremetal provisioning. The reconciliation loop is the most powerful pattern in cloud-native infrastructure.

② **VPC + user-script in Baremetal Server** Automatic network security — for pods and physical servers alike. Security policy is declared once and inherited automatically. No provisioning gap. No manual configuration.

③ **AthenZ integration**

Automatic service identity — get X509 certificate into Baremetal server