THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
KOREA

서울

From Region to Multi AZ

Building Resilient Cloud Infrastructure with OpenStack, Kubernetes, and OVN

서울

Prologue.
kt cloud Universe

About me

Seungjin Han

Cloud Tech Team Leader

kt cloud

Building Cloud Platforms with Open Source

Cloud Platform Engineering

Building KCP, kt cloud's open-source-based cloud platform

OpenStack, Kubernetes, OVN

Multi Region, Multi AZ, Cloud Resilience

Open Source Community Journey

OpenStack Korea Group

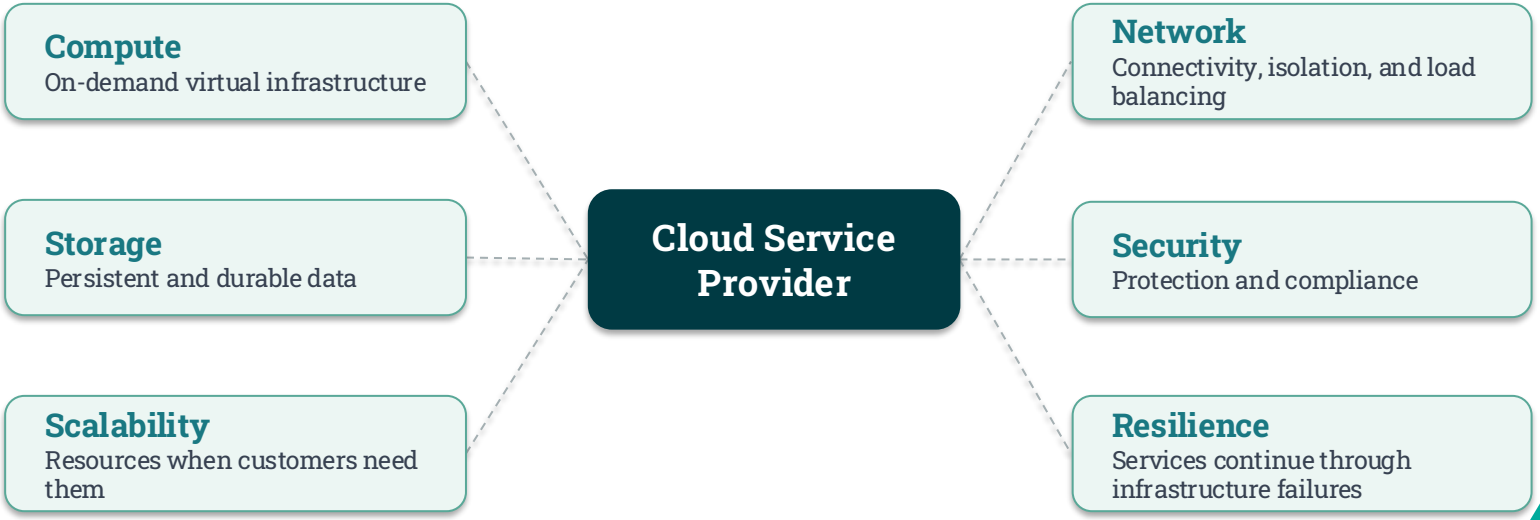
→ Korea Ceph User Group

→ **Kubernetes Korea Group, Organizer**

What Customers Expect from a CSP

Customers consume cloud services.

The CSP takes responsibility for the complexity underneath.



KCP: A New Era for kt cloud

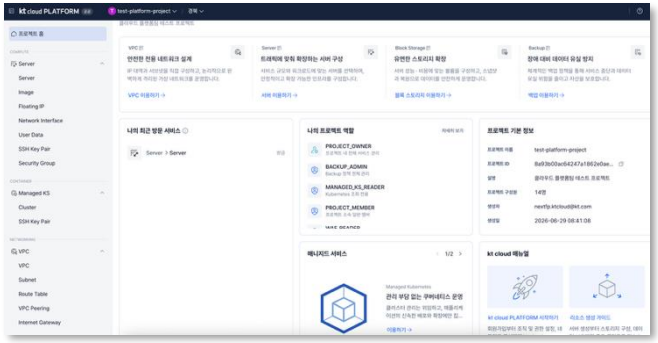
KCP turns open source into a cloud platform built in-house

Open-Source Foundation

Built on proven cloud-native technologies

In-House Engineering

Core platform capabilities developed and operationalized by kt cloud



A new Cloud Platform

Designed, built, and operated by kt cloud – giving us greater control over the platform

Why Korean CSPs Need Greater Platform Control

Local requirements demand more control over infrastructure, operations, and service continuity.

Sovereignty & Data Residency

Keep customer data and critical infrastructure under local control.

Security & Compliance

Meet certification and regulatory requirements with platform-level control.

Service Continuity

Design and operate resilience for critical and public-sector workloads.

Not a Success Story. A Story of Growth

What We Share

The questions, trade-offs and lessons behind the architecture.

What We Hope You Take

A practical perspective – not a prescription.

서울

Part 1.
Disaster Recovery

Disaster Recovery

For a CSP, surviving a datacenter failure is a platform responsibility.

Service continuity is required

Critical cloud services must survive the loss of an entire datacenter.

Recovery becomes the CSP's responsibility

The platform must provide a foundation for restoring services, data, and connectivity.

DR becomes an architectural constraint.

Once a datacenter becomes a failure domain, geography becomes part of the architecture.

Can DR Be Active-Active?

Disaster recovery needs distance. Active-active needs proximity.

DR needs distance

> Disasters can affect an entire geographic area.
>
> The recovery site must be separated from the primary site to avoid the same failure domain.

Active-Active needs proximity

> Clustering, quorum, and synchronous replication require low and predictable latency.
>
> As distance increases, real-time coordination becomes more difficult.
>
> **Greater distance ➔ Higher coordination cost**

서울

Part 2.
Multi-AZ

Why Multi AZ

DR helps us recover. Multi-AZ helps us continue.

DR is designed for recovery

> A distant recovery site protects against large-scale disasters.
>
> But active-standby DR still requires service and data recovery after a failure.

What we wanted instead

> Keep both sites active and continuously synchronized.
>
> Let the platform detect failures and redirect traffic automatically.
>
> **The service should continue – not restart somewhere else.**

Distance Decides the Architecture

DISTANCE → LATENCY → REPLICATION MODEL → RESILIENCE STRATEGY

The farther apart the sites, the more failure isolation we gain—and the harder synchronous coordination becomes.

SINGLE SITE

One datacenter

0 km

RTT: ~0 ms

Everything shares one failure domain.

NO ISOLATION

MULTI-AZ

Yongsan ↔ Mokdong

~30 km

RTT: < 1 ms

Synchronous clustering and quorum are practical.

CONTINUE

MULTI-REGION

Seoul ↔ Gyeongbuk

~200 km

RTT: ~3 ms

Asynchronous replication is the practical choice.

RECOVER

Multi-AZ keeps services running. **Multi-Region** restores them after large-scale failure.

서울

Part 3.
KCP Architecture

Cloud-Native OpenStack

OpenStack provides the cloud. Kubernetes provides the operating model.

OpenStack

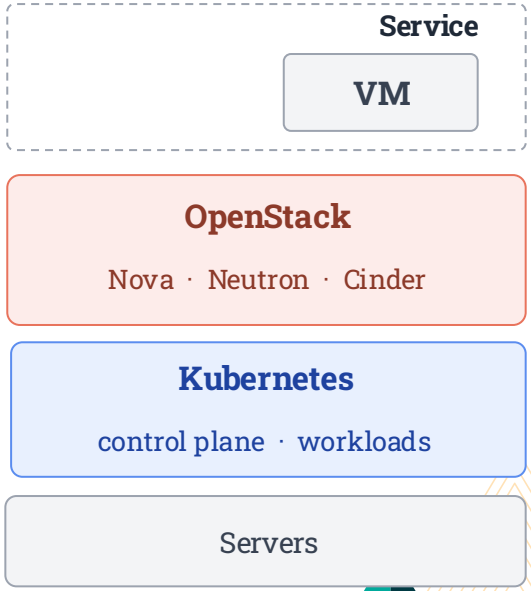
Compute, Network, Storage

Kubernetes

Deployment, Reconciliation, Lifecycle

Automation

Cluster API, GitOps



Two Platforms, Two Network Stacks

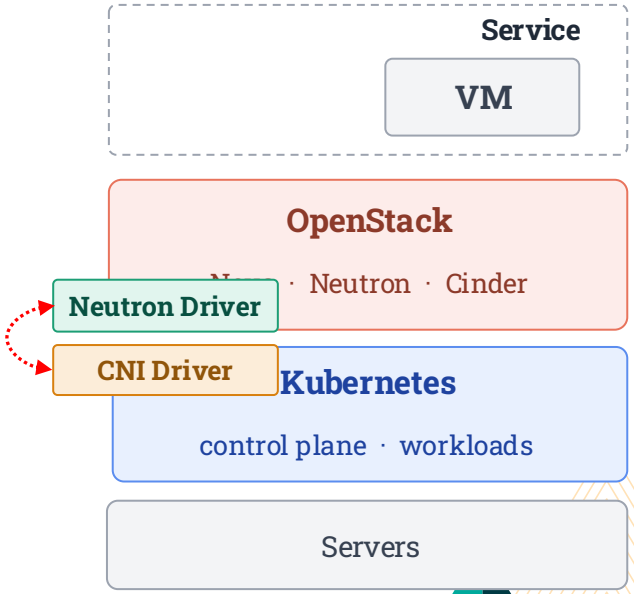
Running OpenStack on Kubernetes does not automatically create one network.

Kubernetes Networking

POD connectivity through CNI

OpenStack Networking

VM and VPC connectivity through Neutron



Separate network stacks increase routing complexity, policy duplication, and troubleshooting overhead.

One Network Model with OVN

Kubernetes and OpenStack share the same networking foundation.

Unified Connectivity

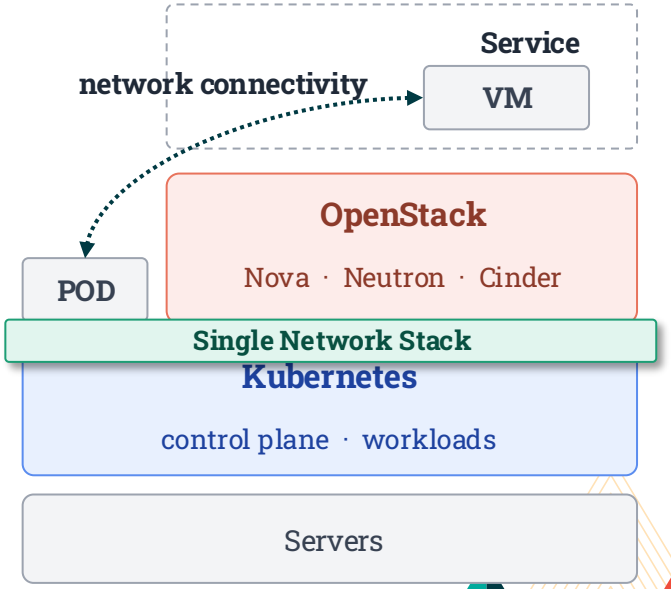
PODs and VMs use a common OVN-based network foundation.

Consistent Policy

Routing and security policies follow the same network model.

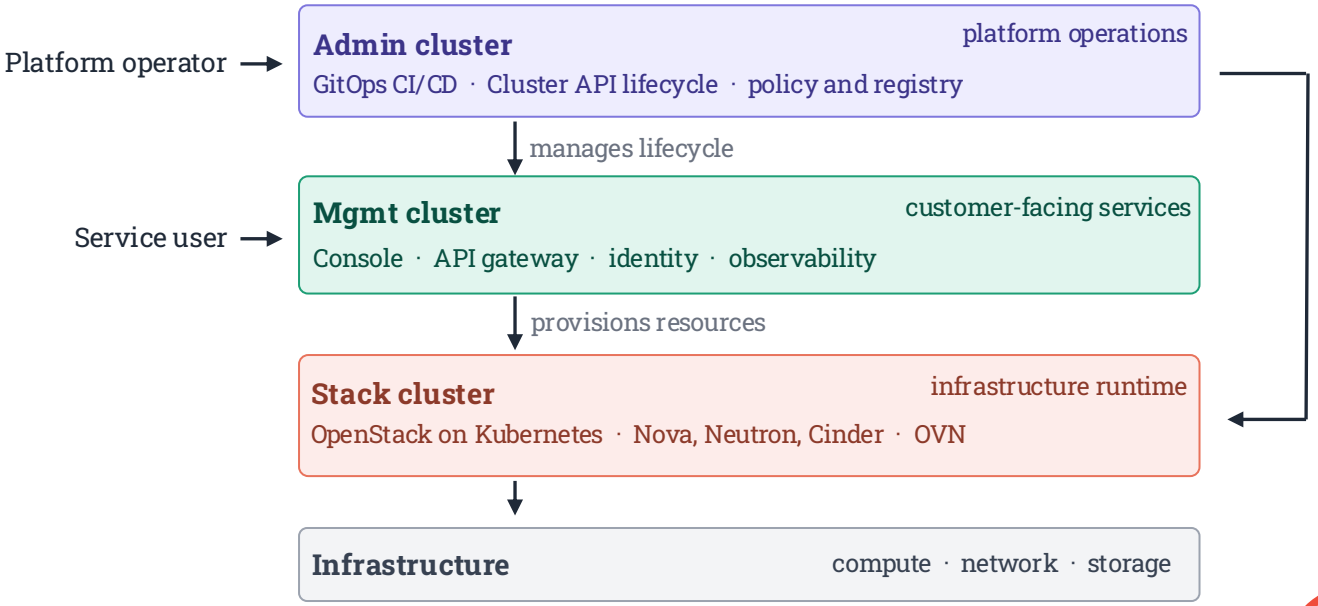
Operational Simplicity

Fewer network boundaries to trace and troubleshoot.



Three Clusters, Three Responsibilities

Separate platform operations, customer services, and infrastructure runtime.



서울

Part 4.
Multi Region Architecture

Multi-Region Means Recovery

Regions are independent by design,

so a regional failure does not become a platform-wide failure.

Independent Regions, Loosely Coupled

Each region operates independently and shares only what must be global.

Global and Regional Services

Not every service needs the same recovery model.

Global

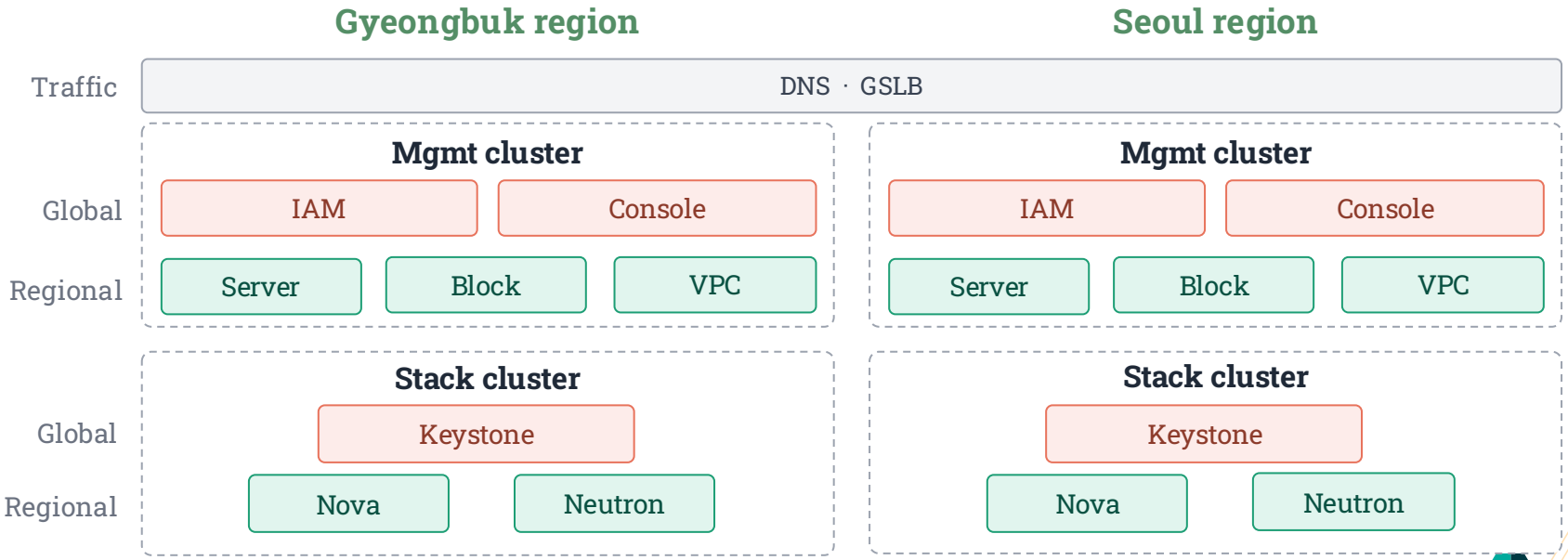
> **One logical service across regions**
>
> Console, IAM, Keystone
>
> Primary / DR model

Regional

> **Independent services per region**
>
> Nova, Neutron, Cinder
>
> Recovered locally when required

What Must Be Global?

What has to be one thing across regions, and what does not.



Keystone: One Identity Across Regions

Keystone is global, but the two regions remain operationally independent.

Single Logical Endpoint

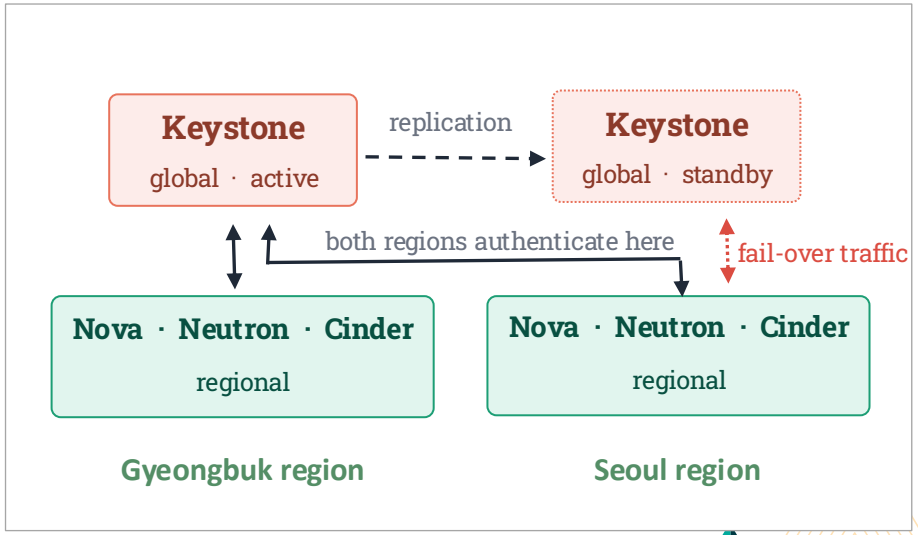
Regional services use the same Keystone endpoint.

DNS-Based Redirection

The internal DNS target changes during failover.

Controlled Failover

Switch only after the DR service and data are validated.



Keystone Failover

Restore the state first. Switch the endpoint only after validation

Step 1 - Restore

Restore the DR MariaDB from the latest validated backup.

Step 2 - Activate

Start and validate Keystone in the DR region.

Step 3 - Redirect

Redirect regional services to the DR Keystone endpoint.

서울

Part 5.
Multi AZ Architecture

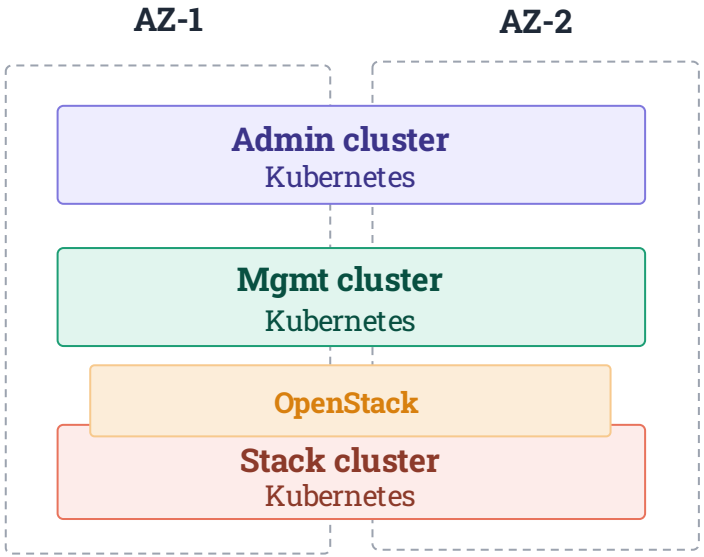
Multi-AZ Means Continuity

A zone failure should be handled by the platform,

not by the customer.

One Cluster Across Two AZs (stretched cluster)

Logically one platform, physically distributed across two failure domains.



Two AZs Need a Third Vote

Two AZs alone are not enough for quorum-based services.

Consensus-based services require a third vote to maintain quorum.

Option 1. Satellite

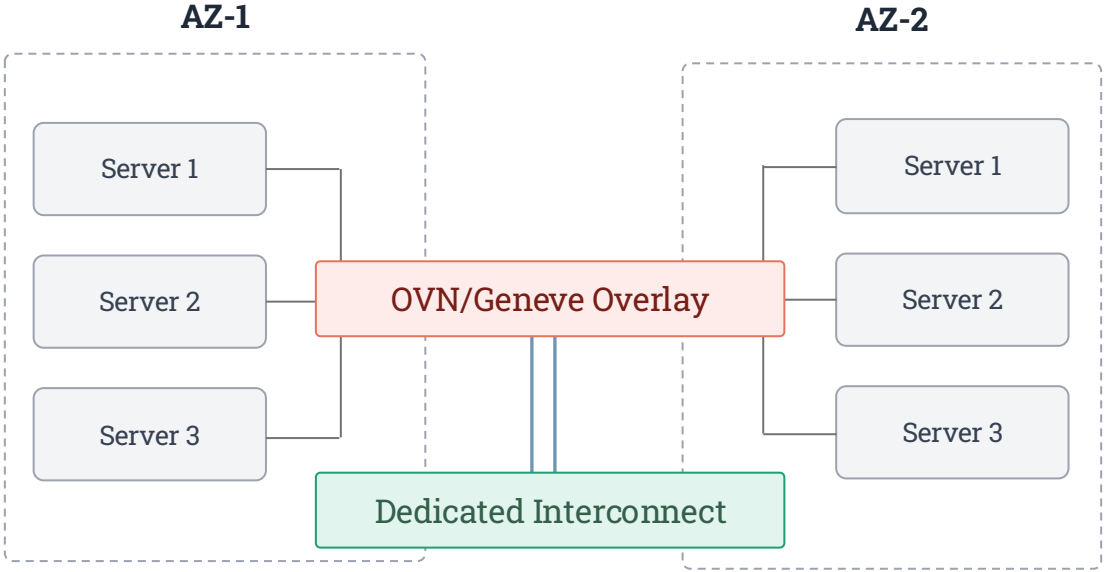
> A minimal third site dedicated to quorum
>
> No production workloads or user traffic
>
> Lower infrastructure cost than building a full third AZ

Option 2. AZ3

> A fully functional third availability zone
>
> Physically separated within the Seoul region
>
> Runs production workloads as a normal AZ

Inter-AZ Connectivity

A stretched cluster depends on predictable, low-latency connectivity between AZs.



Active-Active Traffic Across AZs

Both AZs serve traffic in normal operation. Failure only changes where new traffic goes.

Normal:

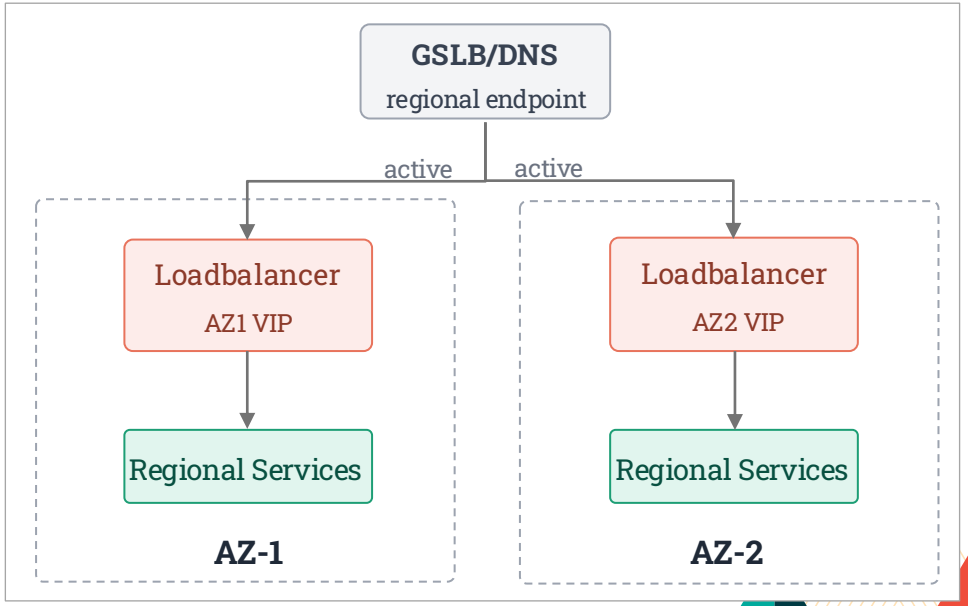
Both AZs receive production traffic

Failure:

The unhealthy AZ is removed from DNS/GSLB responses.

Isolation:

Each AZ maintains its own load balancer and service path.



Keep Workloads Local When Possible

Multi-AZ does not mean every request should cross AZ boundaries.

Compute - Place Locally

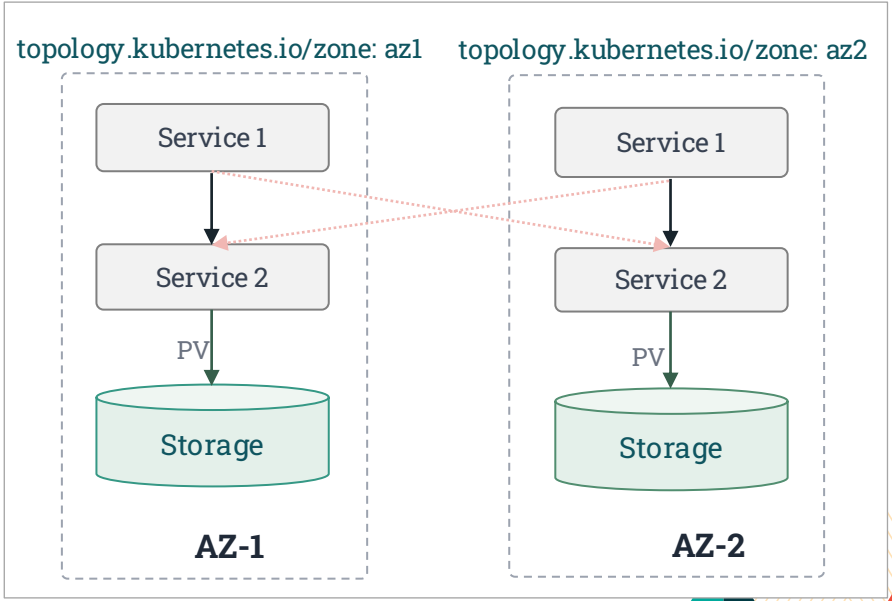
Schedule replicas across AZs while keeping workloads topology-aware.

Network - Route Locally

Prefer same-AZ service endpoints whenever healthy endpoints are available.

Storage - Provision Locally

Create volumes from the storage backend associated with the workload AZ.



Distribute for **resilience**. Stay local for **performance**.

Kubernetes Placement Strategy

Spread by default. Enforce isolation only where required.

Default Policy – Topology Spread Constraints

> Spread replicas across `topology.kubernetes.io/zone`
>
> Also spread replicas across `kubernetes.io/hostname`
>
> `maxSkew: 1`

Workload-Specific Policy – Stricter placement when required

> `whenUnsatisfiable`
>
> Pod anti-affinity
>
> Node affinity

OpenStack Across Availability Zones

OpenStack services must understand the same failure domains as Kubernetes.

Nova – Compute Placement

- Map compute hosts to availability zones
- Default scheduling across available AZs
- Explicit AZ selection when required

Cinder – Storage Locality

- Use separate storage backends for each AZ
- Volume types map workloads to the appropriate AZ

Glance

- Make Image data accessible from both AZs
- Avoid image availability becoming an AZ dependency

OpenStack Across Availability Zones

OpenStack services must understand the same failure domains as Kubernetes.

Neutron / OVN – Network Continuity

> Provide network connectivity across AZs
>
> Gateway and network service placement must account for AZ failures
>
> Maintain connectivity when one AZ becomes unavailable

서울

Resilience Is a Journey,

Not a Feature.

서울

Thanks

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
KOREA