

Broker Deployment Architecture Guide

VPS vs Kubernetes (Kafka / RabbitMQ)

This document explains:

- Should Kafka/RabbitMQ run on same server?
 - VPS architecture best practices
 - Kubernetes architecture best practices
 - When to use separate cluster
 - Production recommendations
-

Problem Context

You have microservices:

- URL Shortener Service
- Click Counter Service
- Analytics Service
- Event Broker (Kafka or RabbitMQ)

The broker handles click events and distributes them to consumers.

Since the broker is **stateful and critical infrastructure**, its placement matters.

VPS Architecture

Should Kafka/RabbitMQ run on same VPS as services?

Development / Small MVP

You can run everything on the same VPS using Docker Compose.

Example:

- 1 VPS
- shortener-service
- counter-service
- analytics-service
- postgres
- redis
- kafka or rabbitmq

This works for:

- Low traffic
 - Testing
 - Early startup phase
-

Production (Recommended Setup)

Kafka or RabbitMQ should run on **separate VPS servers**.

Recommended VPS Production Layout

Option A – Small Production

- VPS 1 → Microservices
 - VPS 2 → Kafka (or RabbitMQ)
 - VPS 3 → Database
 - VPS 4 → Redis (optional)
-

Option B – Proper Kafka Cluster

- VPS 1,2,3 → Kafka Brokers (clustered)
 - VPS 4 → Microservices
 - VPS 5 → Database
-

Why Separate in VPS?

Brokers are:

- Memory heavy
- Disk I/O intensive
- Network intensive
- Stateful systems

If broker and services share same VPS:

- CPU contention
 - Memory spikes
 - Disk bottlenecks
 - Hard to scale independently
 - Higher crash risk
-

Kubernetes Architecture

Now let's answer the important question:

If using Kubernetes, should Kafka/RabbitMQ be in a different cluster?

Small Kubernetes Setup

You CAN deploy everything in the same cluster:

- Namespace: app-services
 - shortener deployment
 - counter deployment
 - analytics deployment
- Namespace: infrastructure
 - kafka (StatefulSet)
 - zookeeper (if required)
 - redis

This is acceptable for:

- Small production
 - Moderate traffic
 - Cost-sensitive setups
-

Large-Scale / Enterprise Setup (Recommended)

For serious production systems:

Kafka or RabbitMQ should be in:

- Separate Kubernetes cluster OR
 - Managed service (Confluent Cloud, AWS MSK, CloudAMQP, etc.)
-

Why Separate Cluster in Kubernetes?

Kafka is:

- Stateful
- Sensitive to network latency
- Requires stable storage
- Requires disk throughput guarantees

If in same cluster as app services:

- App auto-scaling can impact broker performance
- Resource contention possible
- Node pressure may affect brokers
- Harder fault isolation

Recommended Production Architecture

Ideal Cloud Setup

Cluster 1 → Application Cluster

- shortener-service
- counter-service
- analytics-service
- autoscaling enabled

Cluster 2 → Messaging Cluster

- Kafka brokers (StatefulSet)
- Persistent volumes
- Dedicated nodes

Cluster 3 → Database Cluster (optional)

OR Use managed database.

Comparison Table

Scenario	Same Server	Separate Server	Separate Cluster
Local Dev	✓ Yes	✗ Not needed	✗ Not needed
Small VPS	✓ Yes	⚠ Recommended	✗ Not needed
Growing Startup	✗ Avoid	✓ Yes	⚠ Optional
Enterprise	✗ Never	✗ Avoid	✓ Yes

Final Recommendations

If Using VPS

- Development → Same VPS is fine

- Production → Broker on separate VPS
 - High Scale → Kafka cluster (3 brokers minimum)
-

⚙️ If Using Kubernetes

- Small scale → Same cluster, different namespace
 - Medium scale → Dedicated nodes for broker
 - Large scale → Separate cluster OR managed service
-

🛡️ Golden Rule

Stateless services can scale easily.

Stateful infrastructure (Kafka/RabbitMQ, DB, Redis):

- Must be isolated
 - Must have dedicated resources
 - Must scale independently
-

🕒 Final Answer to Your Question

VPS?

Yes — Kafka/RabbitMQ should be on a different VPS in production.

Kubernetes?

For serious production systems — yes, separate cluster (or managed service) is recommended.

For small setups — same cluster is acceptable with proper resource isolation.

You are now thinking like a real distributed systems architect 🧑🏻💻