

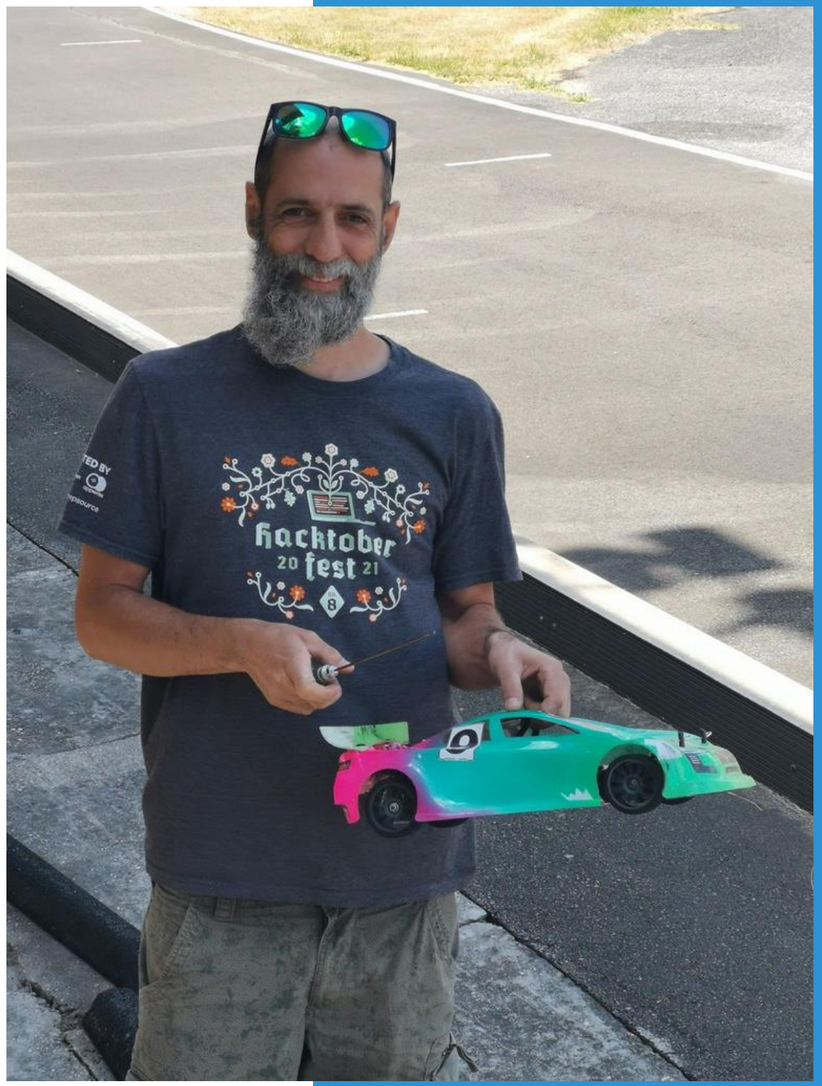
Beyond etcd:

Building a Kubernetes-Compatible State API with Kafka

Richa’rd Kova’cs

Boring Slide
- Always living on the cutting edge

- Founcer, CTO @ HariKube
- mhmxs @ LinkedIn, GitHub
- First Basic program - 1993
- First Linux (Debian) - 1997
- Practising DevOps - 2005
- Go(lang) for food - 2017
- Kubernetes addict - 2018
- Day(s) since i recompiled Kubernetes - 0 /
- OSS devotee



HARIKUBE TEAM MEMEBER

WHAT IS KUBERNETES

BORN AT GOOGLE

Kubernetes was created by Google to manage its own massive-scale applications

OPEN-SOURCED SUCCESS

Google then made it public, and the tech community quickly embraced it

RAPID ADOPTION

Its powerful features and broad support led to its widespread use

INDUSTRY STANDARD

Today, Kubernetes is the go-to way to run applications reliably, anywhere

WHAT ISN’T KUBERNETES



KUBERNETES IS FULL FEATURED

but limited by underlaying database

API server

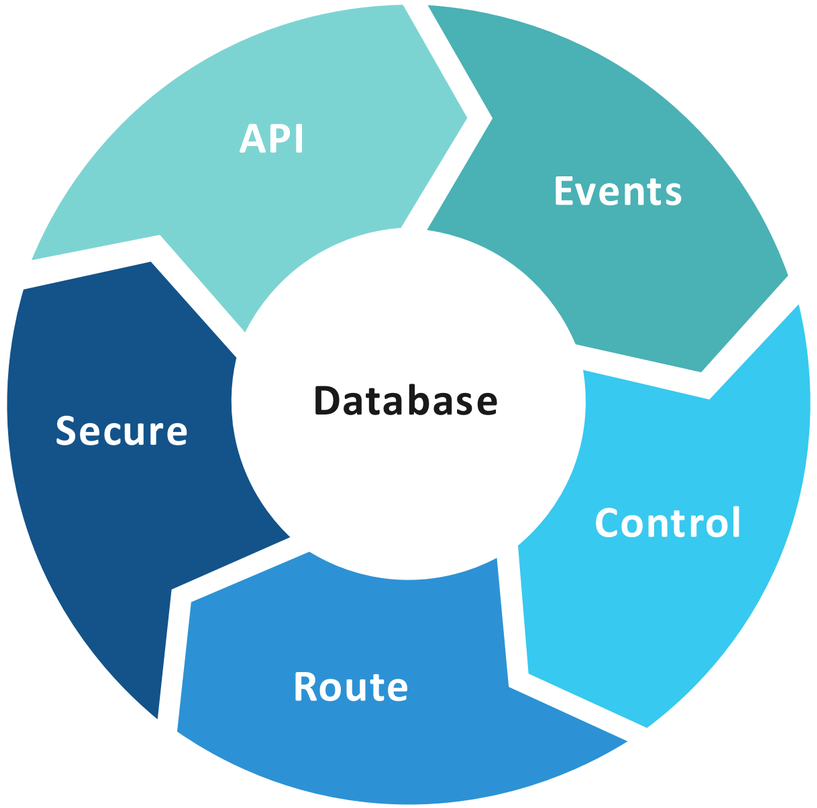
It provides an API server to manage business objects

Event bus

It is useful for real-time event-driven messaging

Controller loop

It executes business logic by reconciling the desired state



Routing traffic

Support of ingress, egress and network policies

Security

Built-in Role-Based Access Control, Service mesh and runtime security

Database

Single-instance database lacks scalability under high load

DATABASE SCALING DIFFICULTIES

“too little, too limited”



Vertical scaling

Vertical scaling is limited by the maximum hardware capacity and becomes cost-prohibitive at scale



Horizontal scaling

Horizontal scaling of the database is ineffective, as it cannot distribute load across multiple database instances



Using SQL database

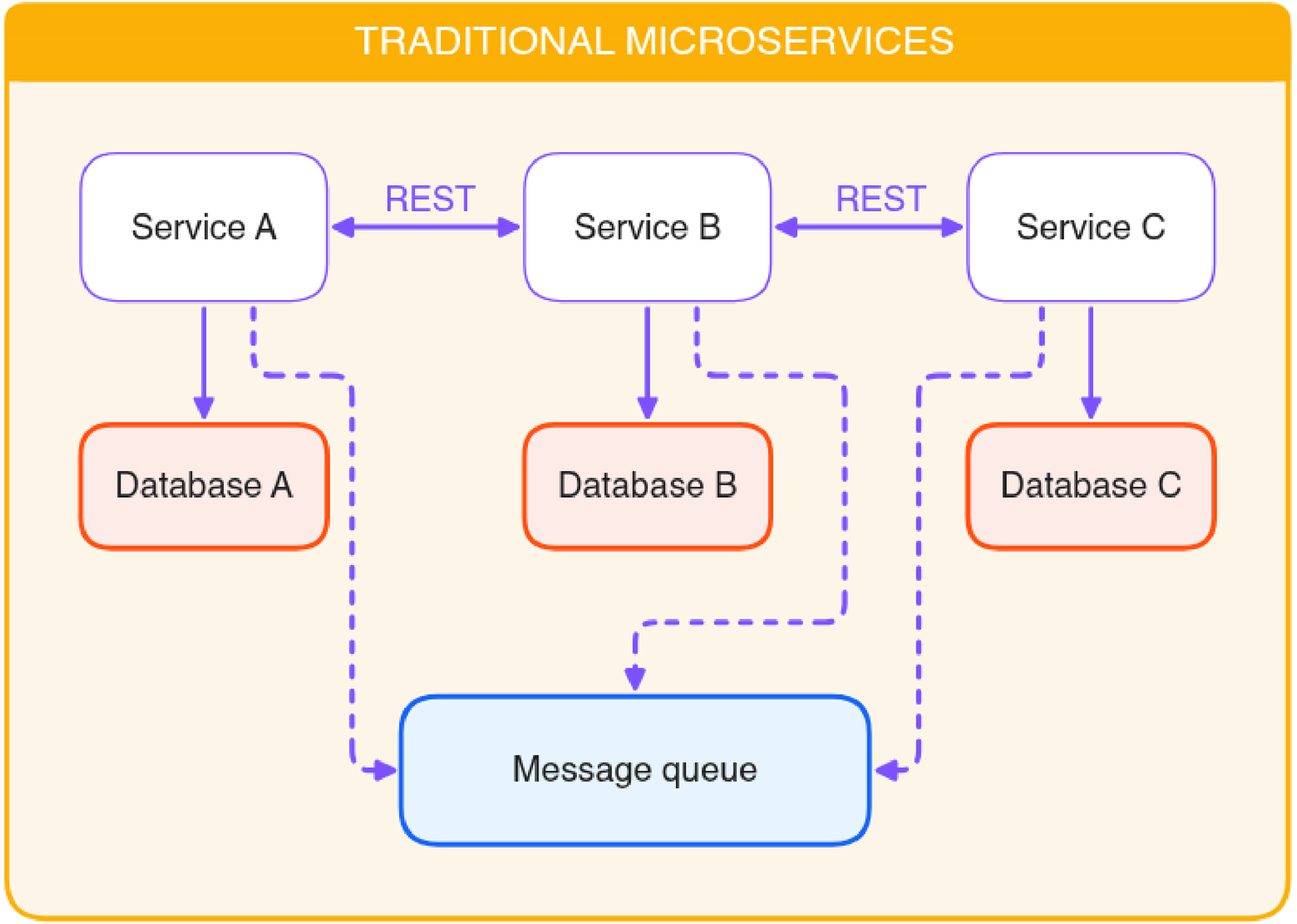
A single-instance database lacks scalability and performance, and Kubernetes API is still a bottleneck



Multi Cluster

Managing complexity, ensuring consistency, and optimizing costs across disparate environments pose significant challenges in hybrid and multi-cloud deployments.

PLATFORM FOR BUILDING PLATFORMS



MEET HARIKUBE



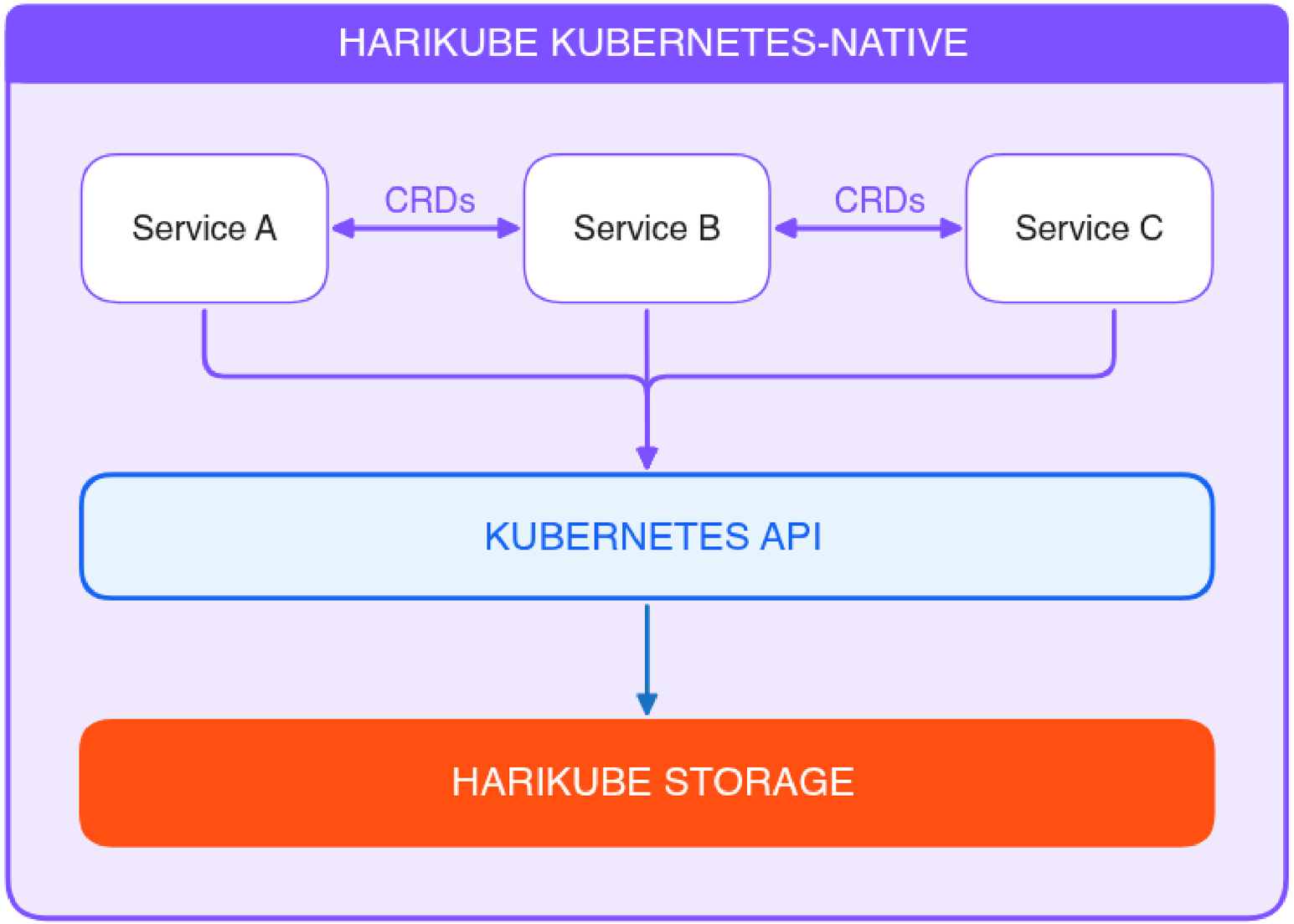
HariKube brings together **Kubernetes**-native API and **Kafka** event streaming in a **petabyte-scale**, versioned **state machine**.

Merge the power of monoliths with the scalability of microservices.

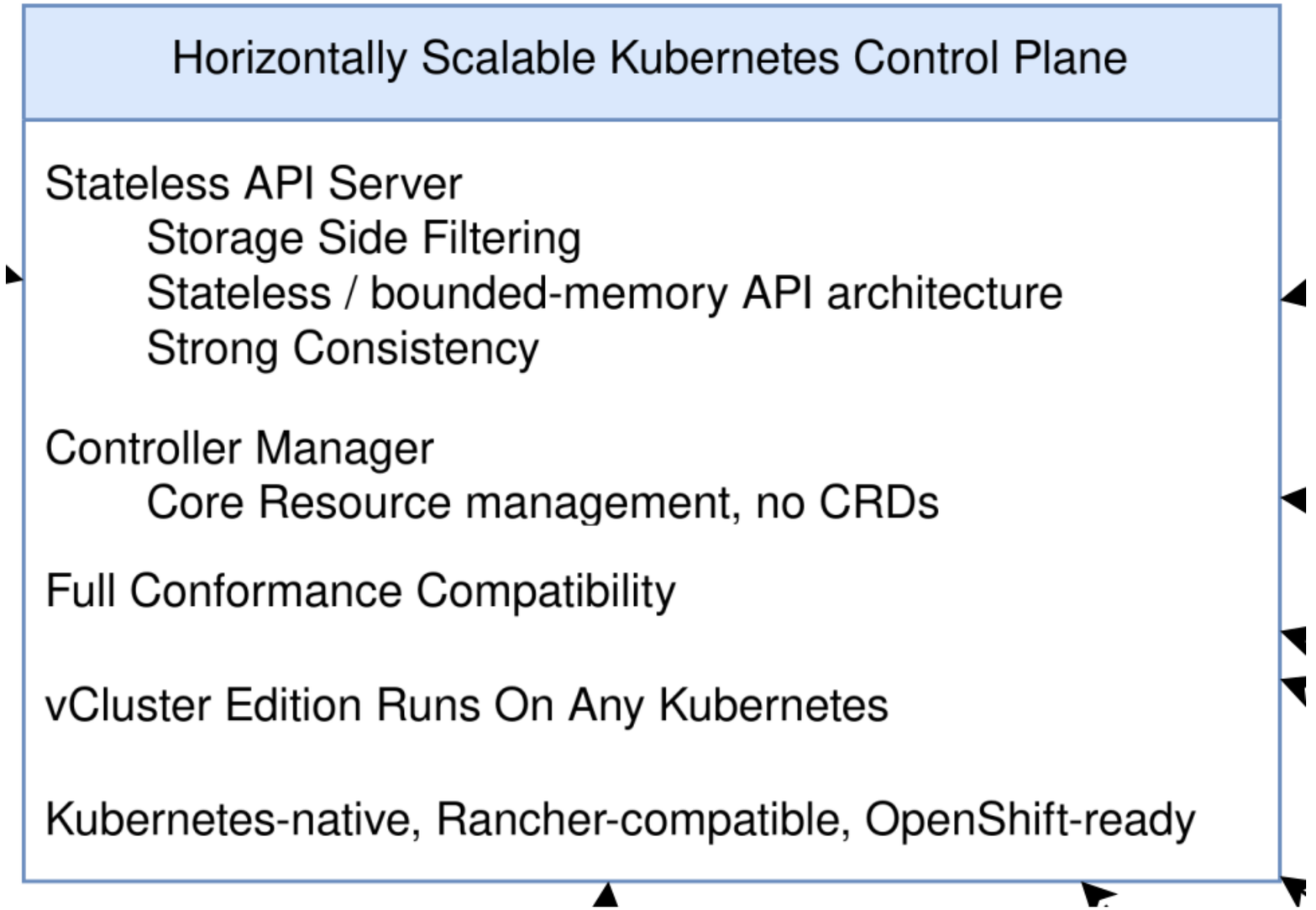
📈 10-50x more throughput
📉 30% less boilerplate
♾️ Full Cloud-Native experience by design

HARIKUBE REDEFINES THE WAY WE THINK ABOUT KUBERNETES

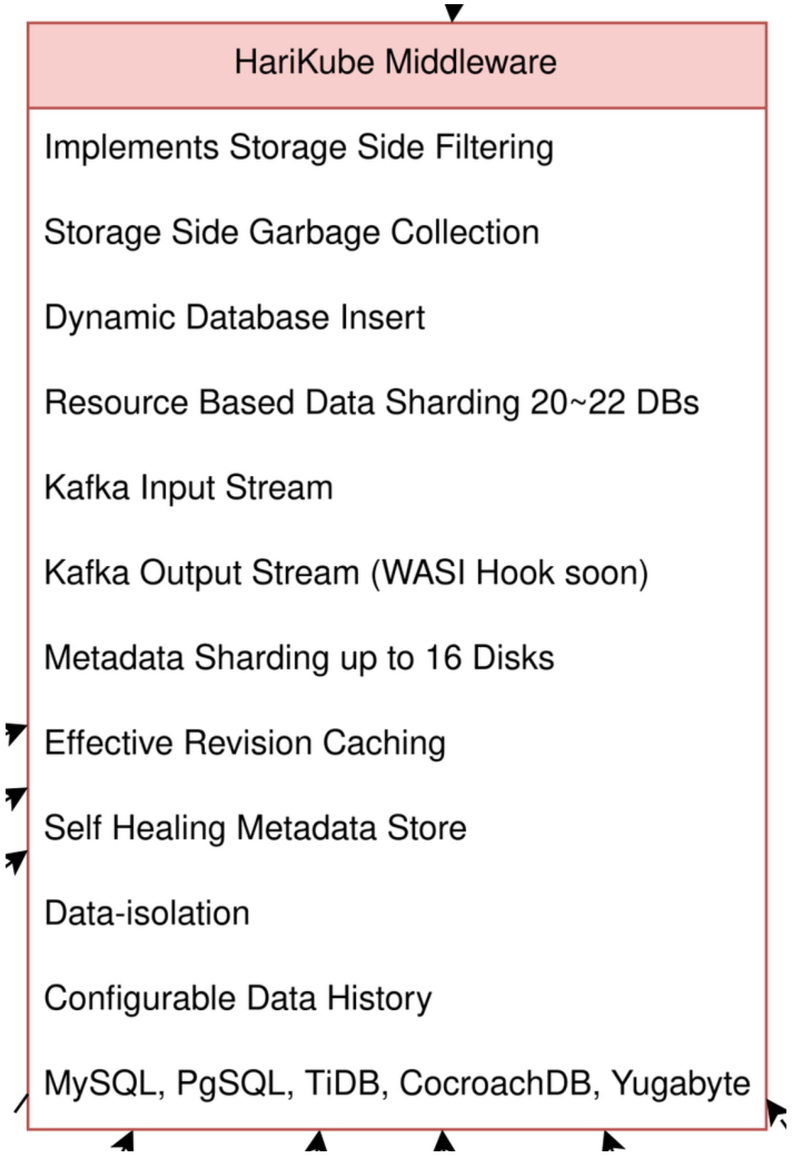
PLATFORM FOR BUSINESS LOGIC



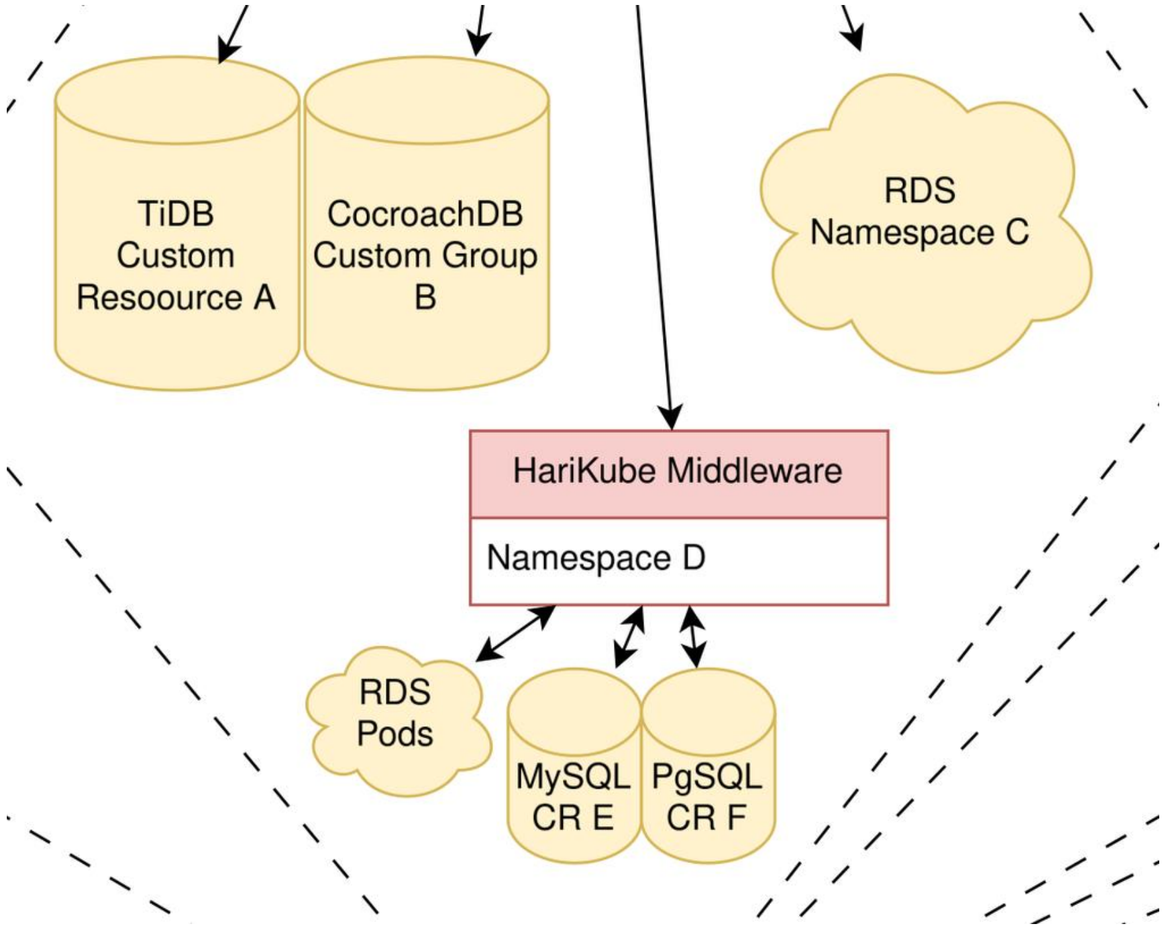
KUBERNETES CONTROLE-PLANE

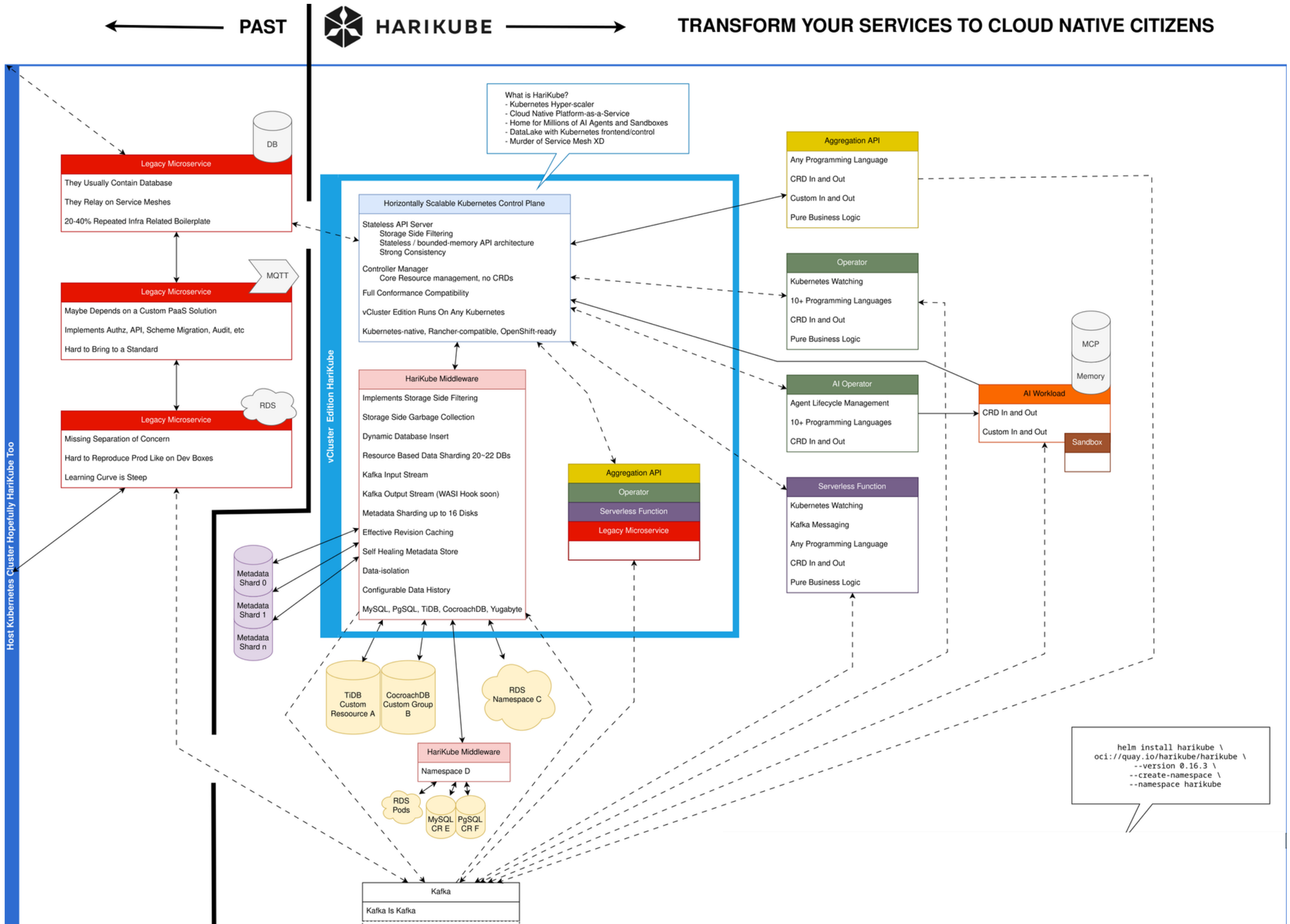


HARIKUBE MIDDLEWARE



DATA LAYER


TiDB
Custom
Resoource A
CocroachDB
Custom Group
B
RDS
Namespace C
HariKube Middleware
Namespace D
RDS
Pods
MySQL
CR E
PgSQL
CR F


PAST
HARIKUBE
TRANSFORM YOUR SERVICES TO CLOUD NATIVE CITIZENS
Host Kubernetes Cluster Hopefully HariKube Too
What is HariKube?
- Kubernetes Hyper-scaler
- Cloud Native Platform-as-a-Service
- Home for Millions of AI Agents and Sandboxes
- DataLake with Kubernetes frontend/control
- Murder of Service Mesh XD
DB
Legacy Microservice
They Usually Contain Database
They Relay on Service Meshes
20-40% Repeated Infra Related Boilerplate
MQTT
Legacy Microservice
Maybe Depends on a Custom PaaS Solution
Implements Authz, API, Scheme Migration, Audit, etc
Hard to Bring to a Standard
RDS
Legacy Microservice
Missing Separation of Concern
Hard to Reproduce Prod Like on Dev Boxes
Learning Curve is Steep
vCluster Edition HariKube
Horizontally Scalable Kubernetes Control Plane
Stateless API Server
Storage Side Filtering
Stateless / bounded-memory API architecture
Strong Consistency
Controller Manager
Core Resource management, no CRDs
Full Conformance Compatibility
vCluster Edition Runs On Any Kubernetes
Kubernetes-native, Rancher-compatible, OpenShift-ready
HariKube Middleware
Implements Storage Side Filtering
Storage Side Garbage Collection
Dynamic Database Insert
Resource Based Data Sharding 20~22 DBs
Kafka Input Stream
Kafka Output Stream (WASI Hook soon)
Metadata Sharding up to 16 Disks
Effective Revision Caching
Self Healing Metadata Store
Data-isolation
Configurable Data History
MySQL, PgSQL, TiDB, CocroachDB, Yugabyte
Metadata Shard 0
Metadata Shard 1
Metadata Shard n
Aggregation API
Operator
Serverless Function
Legacy Microservice
TiDB Custom Resoource A
CocroachDB Custom Group B
RDS Namespace C
HariKube Middleware
Namespace D
RDS Pods
MySQL CR E
PgSQL CR F
Kafka
Kafka Is Kafka
Aggregation API
Any Programming Language
CRD In and Out
Custom In and Out
Pure Business Logic
Operator
Kubernetes Watching
10+ Programming Languages
CRD In and Out
Pure Business Logic
AI Operator
Agent Lifecycle Management
10+ Programming Languages
CRD In and Out
Serverless Function
Kubernetes Watching
Kafka Messaging
Any Programming Language
CRD In and Out
Pure Business Logic
MCP
Memory
AI Workload
CRD In and Out
Custom In and Out
Sandbox
helm install harikube \
oci://quay.io/harikube/harikube \
--version 0.16.3 \
--create-namespace \
--namespace harikube

SAME FEATURES, DIFFERENT SCALE

Metric	HariKube AE	Vanilla K8s	Gain
Throughput	119 req/s ✅	25 req/s ❌	4.8x
Success Rate	100% ✅	100% (then OOM) ❌	not comparable
Latency average	167ms ✅	799ms ❌	4.8x
Latency p95	543ms ✅	2820ms ❌	5.2x
Latency p90	398ms ✅	2470ms ❌	6.2x
Test Duration	60m ✅	~34m (OOM) ❌	not comparable
Stability	Completed ✅	KILLED ❌	not comparable
Objects Handled	200k+ ✅	~26k (crashed) ❌	8x

SAME FEATURES, DIFFERENT SCALE

	HariKube	Vanilla, throughput/memory	Vanilla, latency
Control planes	1 × 4GB = 4GB	5 × 8GB = 40GB	8-20 × 8GB = 64-160GB
Datastore	6 DBs × 3 replicas × 0.5GB = 9GB	5 etcd clusters × 3 members × 1-2GB = 15-30GB (est.)	8-20 etcd clusters × 3 × 1-2GB = 24-120GB (est.)
Objects per cluster	200k	~40k	~10-25k
Total RAM	**~13GB**	**~55-70GB**	**~90-280GB**
200k is...	**Nothing** (a rounding error for 6 SQL DBs)	**The limit** (next step = new cluster)	**Past comfortable**, so you pay with clusters

HARIKUBE IS ...

1. Kubernetes Hyper-scaler
2. Cloud-Native Platform-as-a-Service
3. Home for Large-scale (Autonomous) AI Agents
4. Data-lake with Kubernetes Control-plane
5. State-machine with Real-time Data Streaming
6. Murder of Service Mesh XD

UNIFIED STATE, INDEPENDENT SERVICES

The next architectural step combines a **shared source of truth** with independently running **cloud-native services**

Unified state with cloud-native services
DEV/OPS true separation of concern
3

Microservices with separate data stores
DEVOPS no seperation of concern
2

Monolithic applications
DEV/OPS true seperation of concern
1

Focus more on business logic. Accelerate time to market. Reduce costs.

DEMO



Thank You

https://harikube.com

richard.kovacs@harikube.com