



# KubeBlocks – A domain-driven design operator for stateful workloads



蔡松露/Songlu Tsai

---

ApeCloud.com  
CTO & Cofounder



- 01 Status Quo of stateful workloads on k8s
- 02 What are the real challenges?
- 03 A domain-driven design operator for stateful workloads
- 04 A rich set of day-2 operations is must to have
- 05 Solution for cloud-native applications

Part 1

# Status Quo of stateful workloads on k8s



# Stateful workloads on k8s

---

94% of organizations surveyed deploy services and applications on Kubernetes

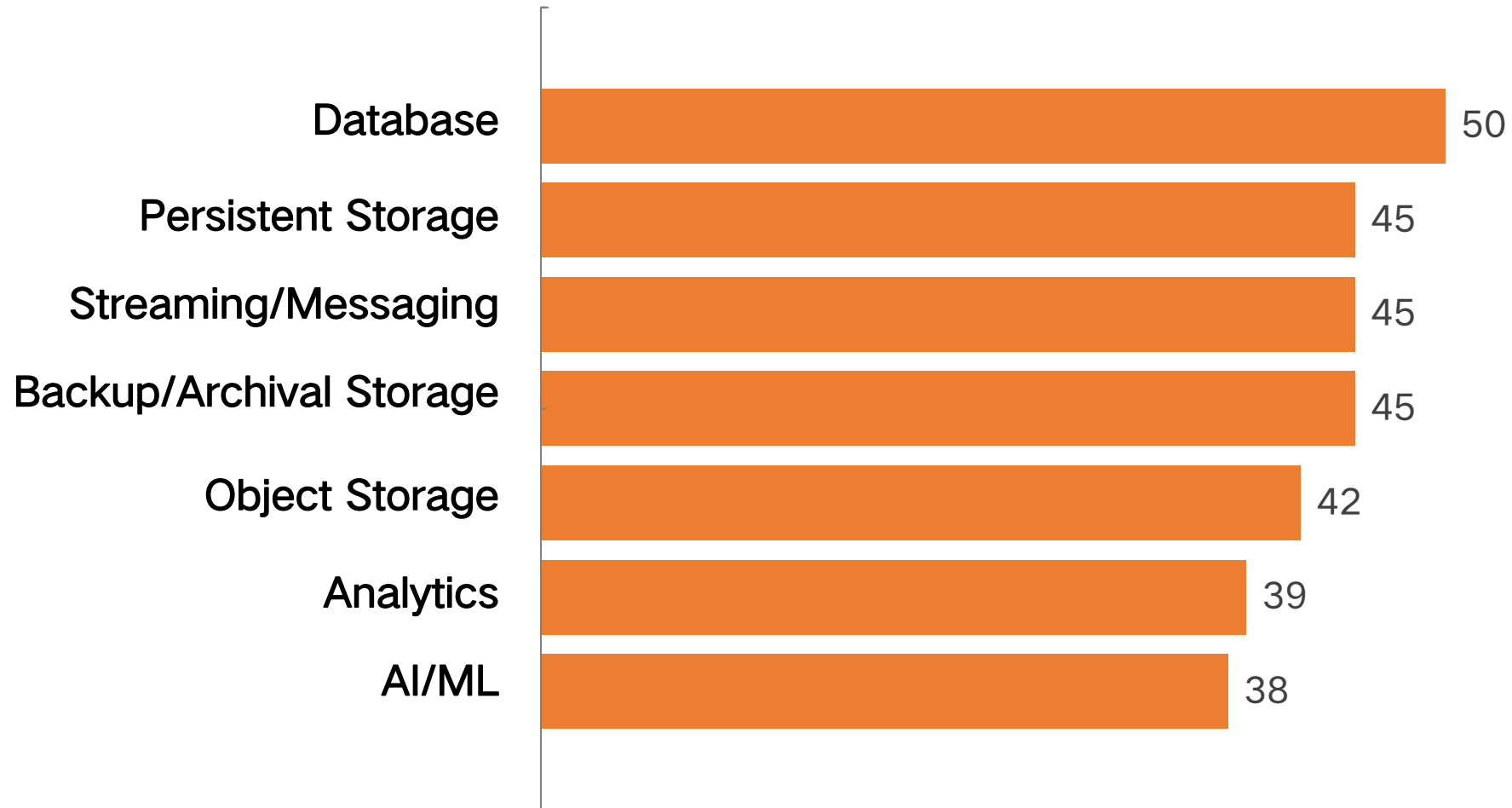
90% of surveyed companies believe Kubernetes is ready for stateful workloads

a large majority (70%) of them are running them in production

Databases take the top spot with persistent storage, streaming, messaging, backup archival storage all tying for the second spot here

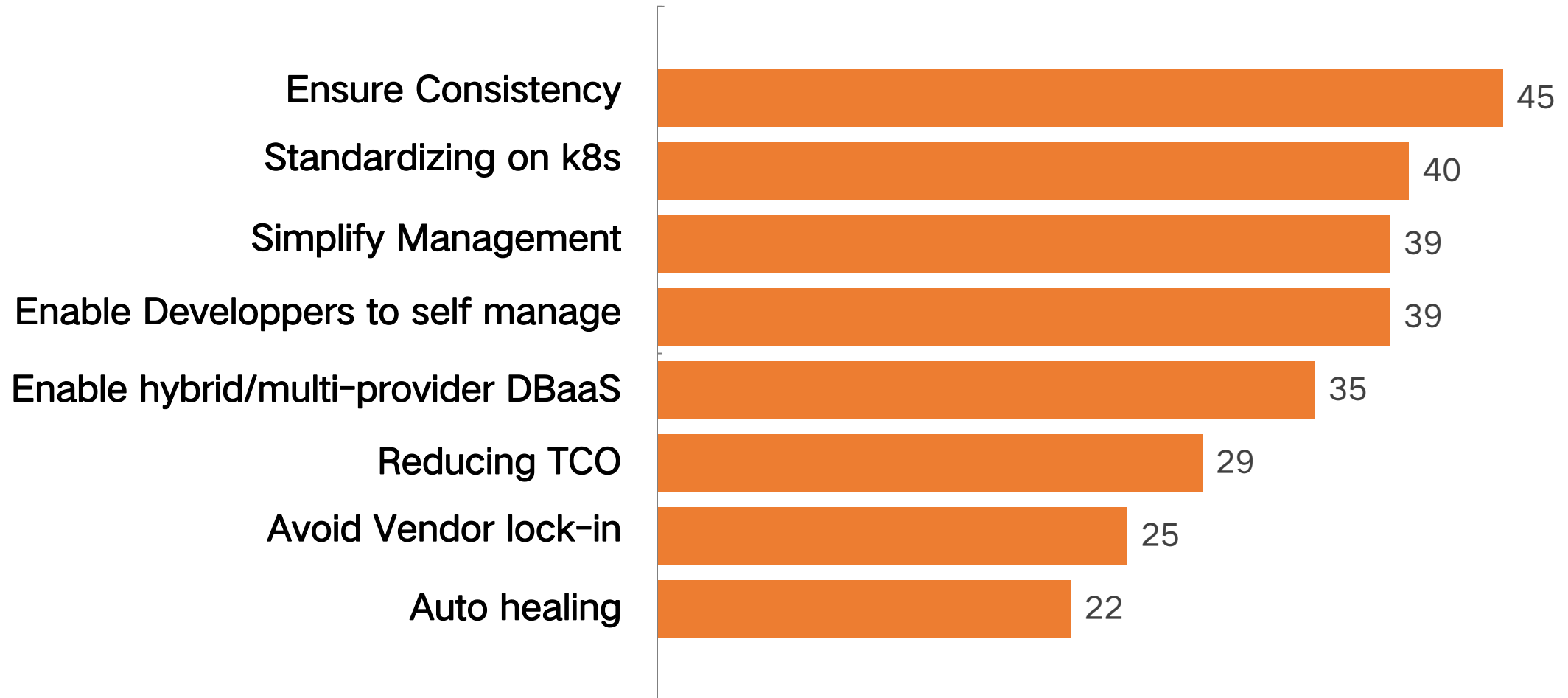
# Stateful workloads on k8s

## Stateful workloads running on k8s



# Stateful workloads on k8s

Most important factors affecting the decision to run stateful workloads on k8s



# Stateful workloads on k8s

## Primary challenges of running data on k8s



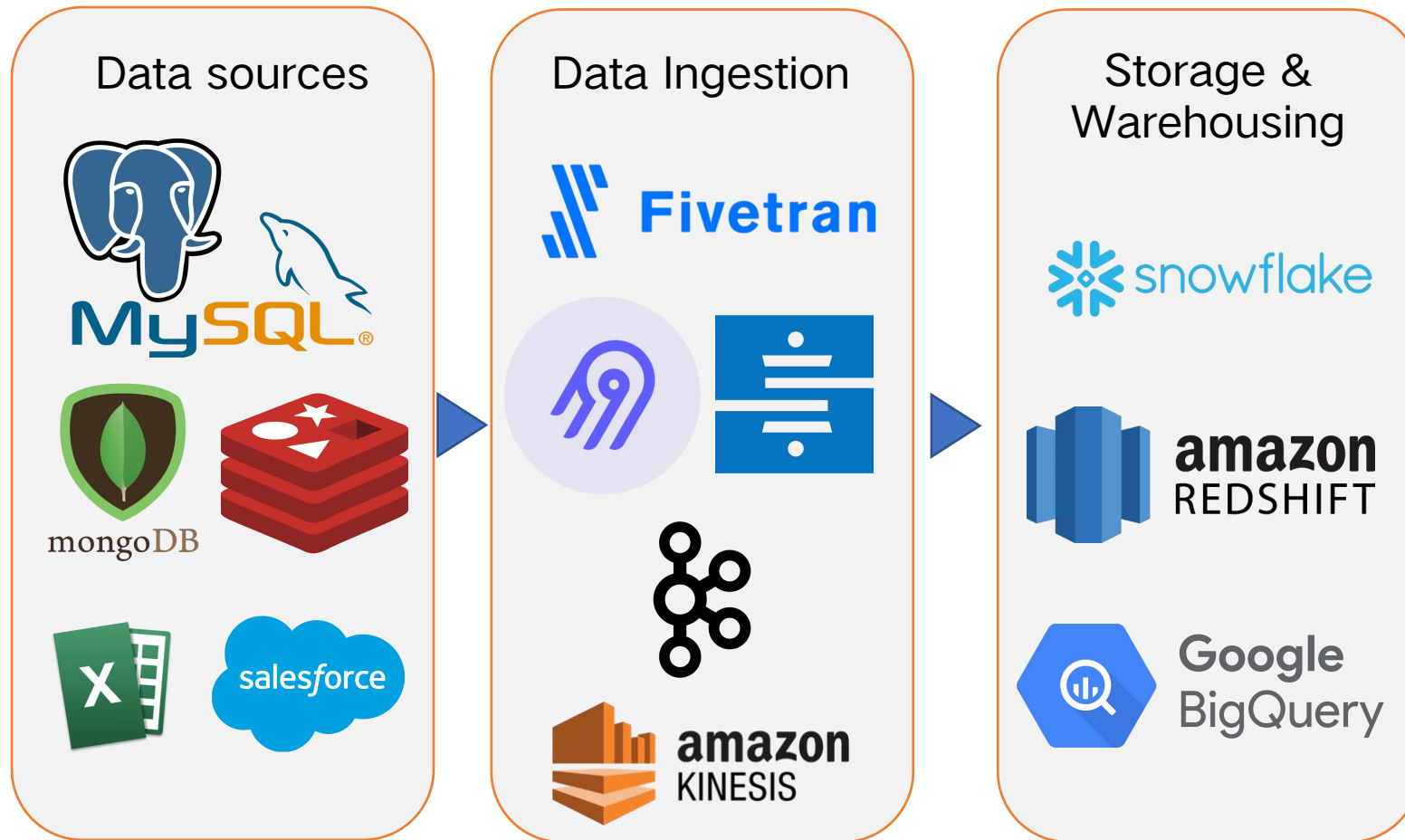
Part 2

# What are the real challenges?



# Multiple databases with Multiple operators

Multiple operators for provisioning a modern data stack



A modern business data flow is composed of:

- OLTP database for transaction
- NoSQL database for caching & k-v retrieving
- Streaming database for ETL
- Warehouse database for data mining

# Existing database operators on k8s

Database	Operator	LifeCycle	Scaling	Backup/Restore	Monitoring	HA
MySQL	Oracle MySQL	Create/Update/Destroy	Scale up	Snapshot	Exporter & Grafana	Group Replication
MySQL	Percona	Create/Update/Destroy	Scale up	xtrabackup	Exporter & Grafana	XtraDB
PostgreSQL	Zalando	Create/Update/Destroy	Scale up	pg_backup	Exporter & Grafana	Patroni
Redis	redis-operator	Create/Update/Destroy	Scale up & out	x	Exporter & Grafana	Sentinel
MongoDB	MongoDB Community	Create/Update/Destroy	Scale up	x	Exporter & Grafana	MongoDB
...	...	...	...	...	...	...

# Operators explosion

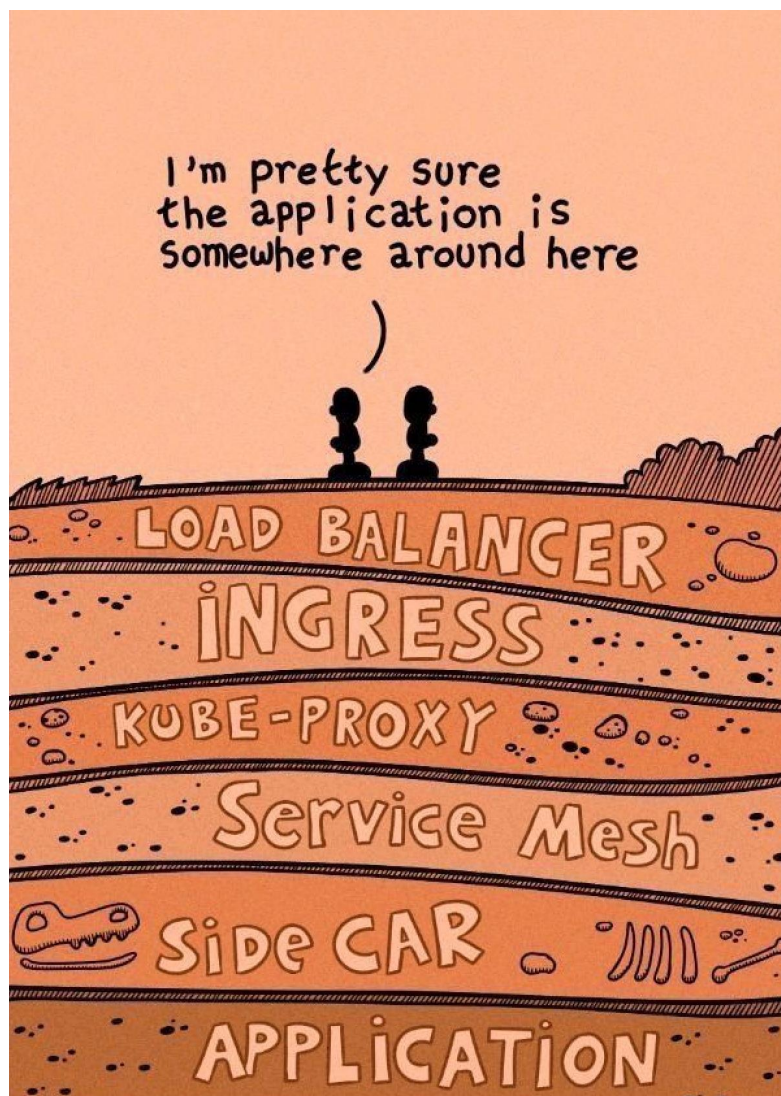
---

All what you need is database, not operators  
K8s is lack of standard for stateful workloads

Multiple operators comes with operators explosion:

- **Inconsistent experience**
  - . different UI & APIs
  - . features are disaligned
- **Learning a bunch of operators**
  - . keep you busy from the real business
- **Maintenance burden**
  - . upgrading becomes disaster
  - . what if an operator stops evolving

# Profound k8s stack



## Trouble shooting is getting harder

In order to troubleshoot an outage, you need to be:

- at least, a profound CKA
  - . or a helm install chokes you
- undoubted, a sophisticated Network Expert
  - . as traffic is magic
- for better, a qualified System Engineer
  - . log is everywhere and nowhere
- at last, a professional DBA
  - . app hangs at web, boss hangs on phone, and you hangs...



# Good at orchestration, but lack of integration

K8s landscape is awesome, but what you need is a necklace, not pearls on beach.

This large grid displays logos for various Kubernetes ecosystem components, organized into several categories:

- Orchestration & Management:** Includes logos for Kubernetes, Crossplane, Volcano, and others.
- Coordination & Service Discovery:** Includes CoreDNS, etcd, and others.
- Remote Procedure Call:** Includes gRPC and others.
- Service Proxy:** Includes Envoy, Contour, and others.
- API Gateway:** Includes Istio, Kong, and others.
- Service Mesh:** Includes Istio, Linkerd, and others.
- Cloud Native Storage:** Includes Rook, Ceph, and others.
- Container Runtime:** Includes cri-o, containerd, and others.
- Cloud Native Network:** Includes Cilium, Calico, and others.
- Automation & Configuration:** Includes Ansible, Terraform, and others.
- Container Registry:** Includes Harbor, Docker Registry, and others.
- Security & Compliance:** Includes Falco, Aqua, and others.
- Key Management:** Includes HashiCorp Vault, and others.

This grid displays logos for various Observability and Analysis tools, organized into several categories:

- Monitoring:** Includes Prometheus, Grafana, and others.
- Logging:** Includes Fluentd, ELK Stack, and others.
- Tracing:** Includes Jaeger, Zipkin, and others.
- Chaos Engineering:** Includes Chaos Mesh, Litmus, and others.
- Continuous Optimization:** Includes Argo Rollouts, and others.

# When you fight with database on k8s

---



Part 3

# A domain-driven design for stateful workloads operators



# Standard is always the first thing

---

As operators have similar features such as lifecycle management, backup, monitoring, upgrade, etc...

Is it possible to build a general purpose CRD and Controller for all stateful workloads ?



# Insights from existing database operators

---

Let's dive into stateful workloads and popular database operators, we can see the facts:

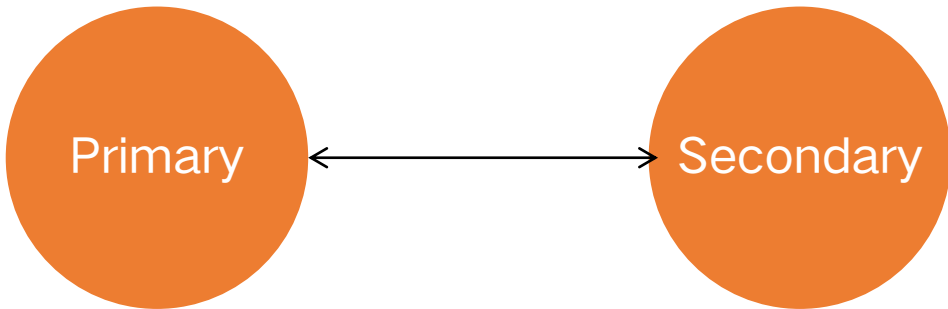
- 1, Major part of the operators is handling with **Lifecycle**(create/destroy/update),while **lack of day-2 operations**
- 2, The stateful workload has **more than one roles/components** in an instance, such as primary, secondary, leader, follower and learner, etc.
- 3, Lifecycle is about processing the **topology and traits** of an instance, the **topology is relations & dependencies** among roles/components, while the **traits are runtime metas** can be mainly described by **podSpec**
- 4, A stateful workload has **many versions against same topology**
- 5, The data **dependencies between components** can be concluded as **primary-secondary** for classic hot **standby** cluster, **leader-follower** for distributed clusters based on **PAXOS or RAFT**



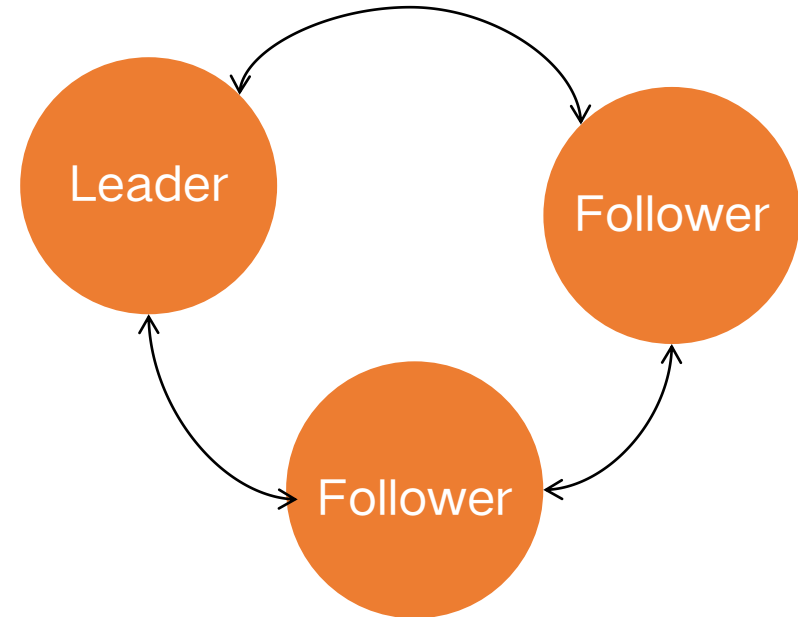
# ReplicationSet & ConsensusSet

A primary-secondary group is so cohesive that we can treat them as a basic set named **ReplicationSet**, while we call the leader-follower group as **ConsensusSet**. ReplicationSet & ConsensusSet are basic **blocks** to build larger stateful clusters. That's also why we got the name **KubeBlocks**.

**ReplicationSet**  
MySQL/PG/Redis/...

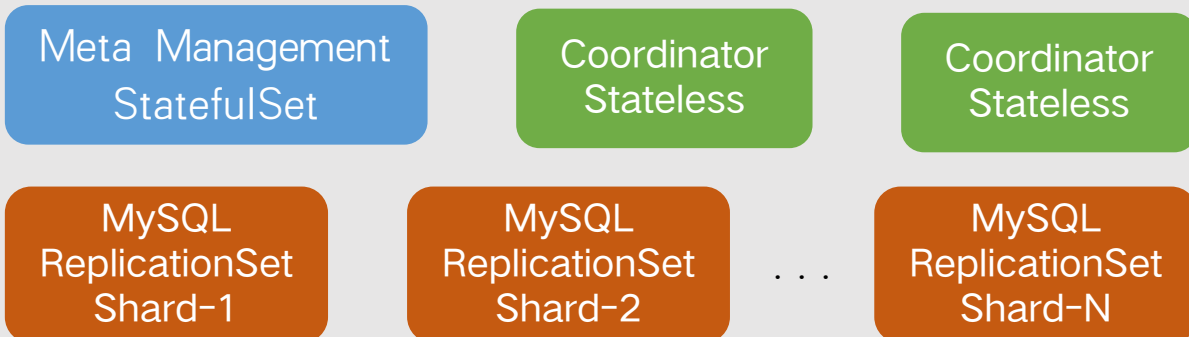


**ConsensusSet**  
ETCD/Zookeeper/MongoDB

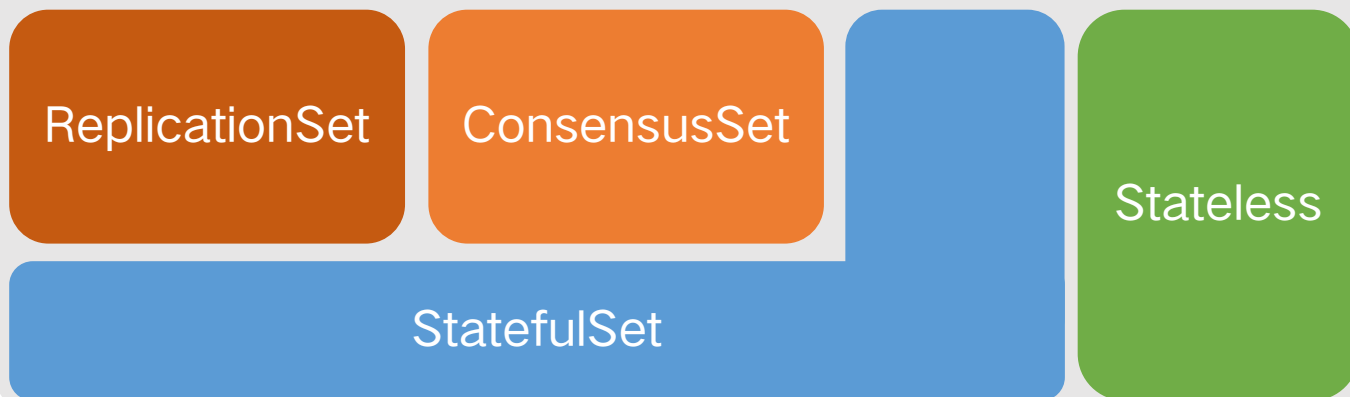


# Basic blocks of KubeBlocks

## A typical MySQL sharding Topology



## Four types of Blocks



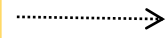
- **Stateless Block** is native k8s stateless deployment, used for proxy, control plane process
- **Statefulset Block** is native k8s statefulset, used for process holding persistent data and providing service alone
- **ReplicationSet Block** is built on StatefulSet, used for classical hot standby cluster
- **ConsensusSet Block** is built on StatefulSet, used for Paxos or RAFT group

# CRDs in KubeBlocks

## ClusterDefinition

- Specification for **topology** and default runtime metas

```
kind:ClusterDefinition
metadata:
spec:
  componentDefs:
  - characterType: postgresql
    workloadType: Replication
  configSpecs:
  logConfigs:
  monitor:
  name: postgresql
  podSpec:
  containers:
```



## ClusterVersion

- Specification for image **version**
- Override the default images in ClusterDefintion

```
kind:ClusterVersion
metadata:
spec:
  clusterDefinitionRef: postgresql
  componentVersions:
  - componentDefRef: postgresql
  versionContext:
  containers:
  - image: spilo:12.14.1
  initContainers:
  - image: spilo:12.14.1
```

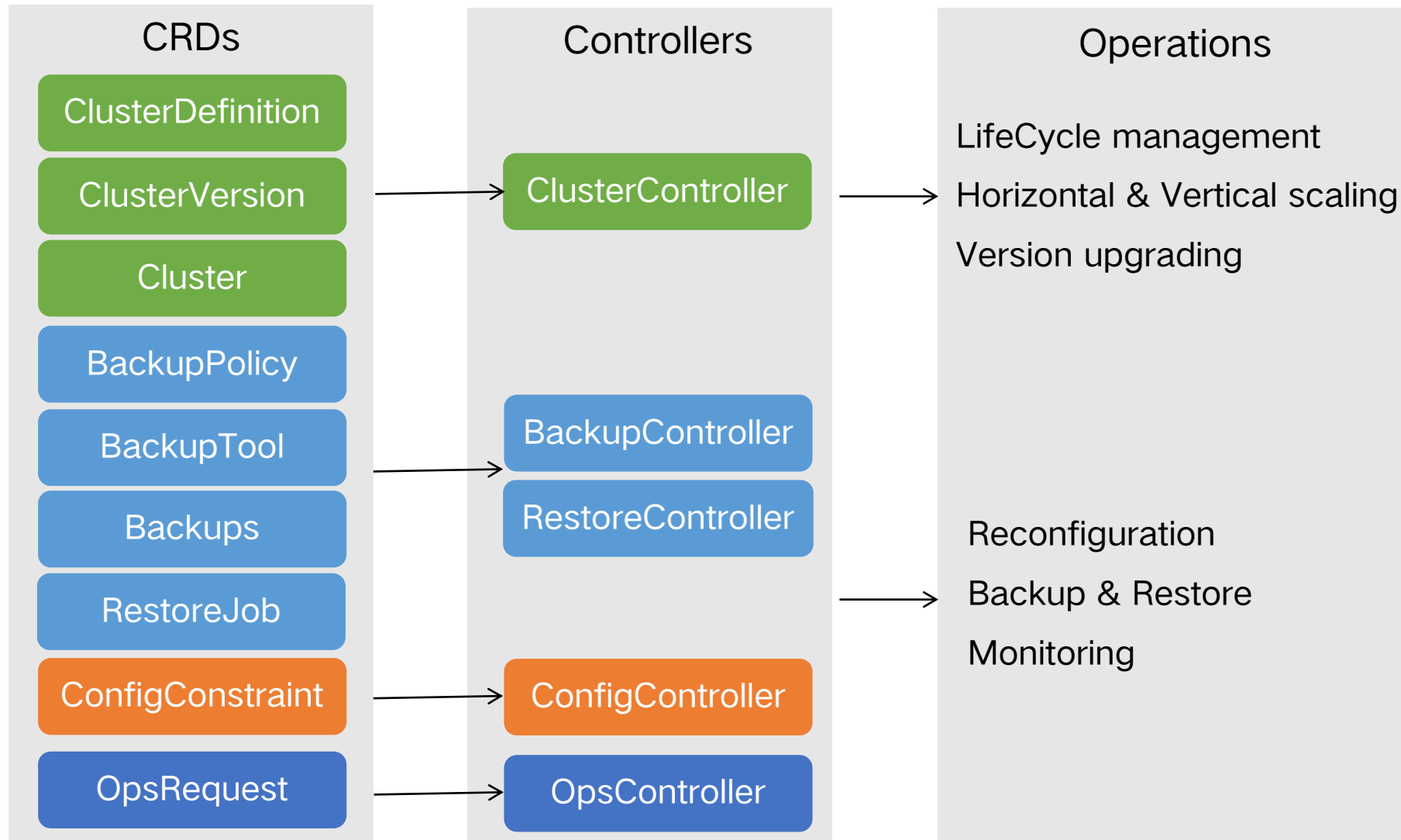


## Cluster

- Specification for a concrete cluster
- Override the default **runtime settings** of monitoring, logs, resources, affinity, VCTs...

```
kind:Cluster
metadata:
spec:
  clusterDefinitionRef: postgresql
  clusterVersionRef: postgresql-12.14.1
  componentSpecs:
  - componentDefRef: postgresql
    replicas: 2
  volumeClaimTemplates:
```

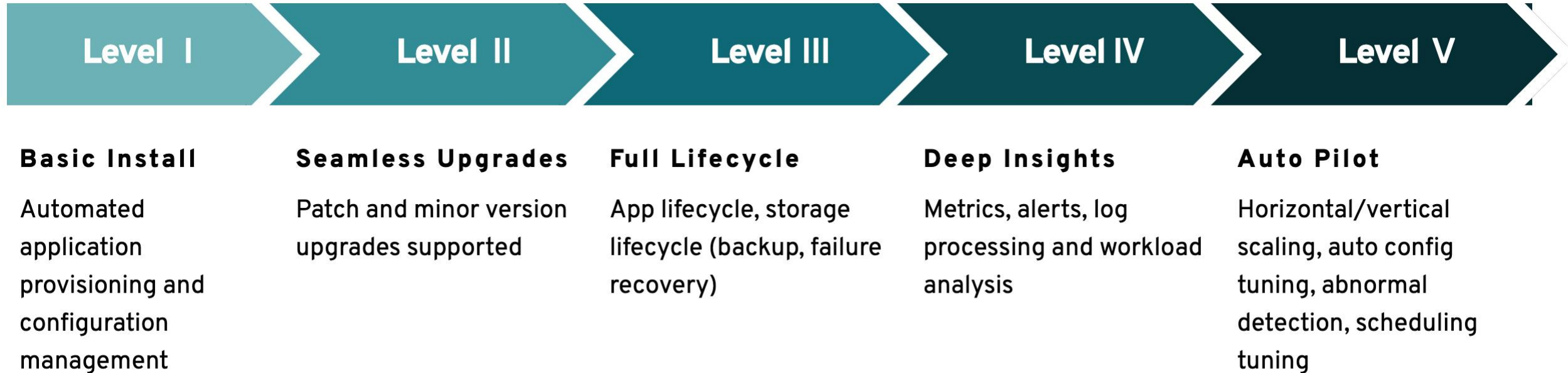
# Controllers in KubeBlocks



Part 4

**A rich set of day-2 operations is  
must to have**

# Operator capability levels



# Reconfiguration

---

```
# 'rose35' is name of a primary-secondary postgresql
# describe the config of cluster rose35
$ kbcli cluster describe-config rose35

# edit the postgresql.conf like kubectl edit
$ kbcli cluster edit-config rose35

# explain the config parameters
# kbcli cluster explain-config rose35

# set max connections to 500
kbcli cluster configure rose35 --set max_connections=500
```



# Backup & Restore

---

```
# create a backup
```

```
kbcli cluster backup rose35
```

```
# create a snapshot backup, make sure the CSI support it
```

```
kbcli cluster backup rose35 --type snapshot
```

```
# create a backup with specified backup policy
```

```
kbcli cluster backup rose35 --backup-policy <backup-policy-name>
```

```
# restore a new cluster from a backup
```

```
kbcli cluster restore new-cluster-name --backup backup-name
```

```
# restore a new cluster from point in time
```

```
kbcli cluster restore new-cluster-name --restore-to-time "Apr 13,2023 18:40:35  
UTC+0800" --source-cluster rose35
```

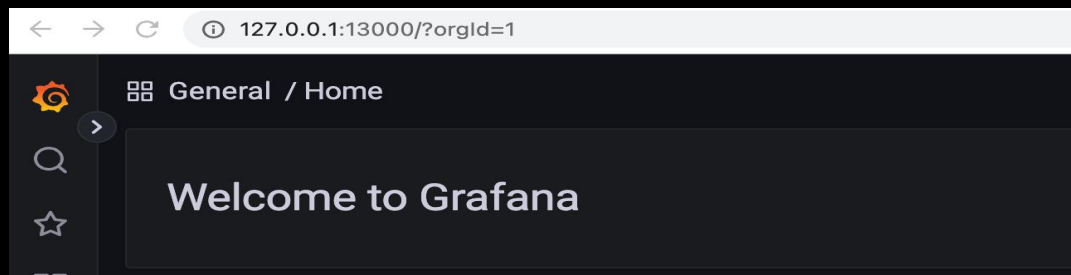


# Monitoring

```
# dashboard of grafana, prometheus and alertmanager
$ kbcli dashboard list
```

NAME	NAMESPACE	PORT	CREATED-TIME
kubeblocks-grafana	kb-system	13000	May 31, 2023 16:45 UTC+0800
kubeblocks-prometheus-alertmanager	kb-system	19093	May 31, 2023 16:45 UTC+0800
kubeblocks-prometheus-server	kb-system	19090	May 31, 2023 16:45 UTC+0800

```
# open grafana in browser
$ kbcli dashboard open kubeblocks-grafana
```



# Monitoring

General / PostgreSQL

namespace	cluster	instan
default	rose35	rose35-postgr
default	rose35	rose35-postgr
default	rose35	rose35-postgr
default	rose35	rose35-postgr
default	rose35	rose35-postgr

Connections (2 panels)

Tuples (5 panels)

Queries (6 panels)

Transactions & WAL (4 panels)

Conflicts & Locks (4 panels)

Buffers & Blocks Operations (7 panels)

Temp files (2 panels)

Database Size (1 panel)

Replication (5 panels)

General / Node Exporter

CPU Cores: 16

RAM Total: 62 GiB

SWAP Total: 0 B

Disk Total: 39 GiB

CPU Busy: 96.2%

RAM Used: 34%

SWAP Used: 0%

Disk Used: 20.5%

Memory

3.73 GiB

1.86 GiB

0 B

RAM Total, RAM Used, RAM Cache + Buffer, RAM Free, SWAP Used

Disk R/W Data

400 MB/s

200 MB/s

0 B/s

nvme1n1 - Read bytes: 26.4 MB/s, 26.9 MB/s

nvme0n1 - Written bytes: 62.7 kB/s, 94.2 kB/s

nvme1n1 - Written bytes: 290 MB/s, 329 MB/s

Disk Space Used

100%

50%

0%

Network Traffic

2 Gb/s

0 b/s

-2 Gb/s

recv cali7e36e8e90d3, recv dummy0, /run/containerd/io.containerd.grpc.v1.cri/sandbox, recv eth0, recv eth1, recv kube-ipvs0, recv lo, recv nodelocaldns

System Uptime: 4.8 hour

System Load: 30.7

CPU

75.00%

50.00%

25.00%

0.00%

22:50 22:55 23:00 23:05 23:10 23:15

Busy System, Busy User, Busy IOWait, Busy IRQs, Busy Other

CPU per Core

CPU Modes

100.0%

50.0%

0.0%

22:50 22:55 23:00 23:05 23:10 23:15

User - Normal processes executing in user mode, Idle - Waiting for something to happen

CPU System Time

200.0 ms

0 s

22:50 22:55 23:00 23:05 23:10 23:15

	Mean	Last *	Max	Min
CPU 0	122 ms	160 ms	210 ms	0 s
CPU 1	103 ms	120 ms	200 ms	0 s

CPU Idle Time

1.0 s

500.0 ms

0 s

22:50 22:55 23:00 23:05 23:10 23:15

	Mean	Last *	Max	Min
CPU 0	247 ms	20.0 ms	980 ms	10.0 ms
CPU 1	273 ms	40.0 ms	1 s	10.0 ms

# Horizontal & vertical scaling

---

```
# horizontal scaling of rose35, add a replica for cluster  
$ kbcli cluster hscale rose35 --replicas=3
```

OpsRequest rose35-horizontalscaling-ldlwk created successfully, you can view the progress:

```
-- kbcli cluster describe-ops rose35-horizontalscaling-ldlwk -n default
```

```
# vertical scaling of cpu and memory
```

```
$ kbcli cluster vscale rose35 --components=postgresql --cpu=500m --  
memory=500Mi
```

# Horizontal & vertical scaling

```
slc@slcmac kubeblocks % kbcli cluster describe-ops rose35-horizontalscaling-ldlwk -n default
Spec:
  Name: rose35-horizontalscaling-ldlwk  NameSpace: default      Cluster: rose35 Type: HorizontalScaling

Command:
  kbcli cluster hscale rose35 --components=postgresql --replicas=3 --namespace=default

Last Configuration:
COMPONENT  REPLICAS
postgresql      2

Status:
  Start Time:      Jun 06,2023 12:48 UTC+0800
  Completion Time: Jun 06,2023 12:49 UTC+0800
  Duration:        50s
  Status:          Succeed
  Progress:        1/1

  OBJECT-KEY      STATUS  DURATION  MESSAGE
  Pod/rose35-postgresql-2  Succeed  36s      Successfully created pod: Pod/rose35-postgresql-2 in Component: postgresql

Conditions:
LAST-TRANSITION-TIME  TYPE          REASON                      STATUS  MESSAGE
Jun 06,2023 12:48 UTC+0800  Progressing   OpsRequestProgressingStarted  True    Start to process the OpsRequest: rose35-horizontalscaling-ldlwk in Cluster: rose35
Jun 06,2023 12:48 UTC+0800  Validated    ValidateOpsRequestPassed      True    OpsRequest: rose35-horizontalscaling-ldlwk is validated
Jun 06,2023 12:48 UTC+0800  HorizontalScaling  HorizontalScalingStarted      True    Start to horizontal scale replicas in Cluster: rose35
Jun 06,2023 12:49 UTC+0800  Succeed      OpsRequestProcessedSuccessfully  True    Successfully processed the OpsRequest: rose35-horizontalscaling-ldlwk in Cluster: rose35
```



# Version upgrading

---

# Attention : the major version upragde may cause a failure due to incompatible data format, so we suggest that the minor version upgrade through the 'upgrade' subcommand, the major version upragde through a data migration

# upgrade the cluster to the target version

```
kbcli cluster upgrade rose35 --cluster-version=postgresql-14.7.2
```

OpsRequest rose35-upgrade-6bwct created successfully, you can view the progress:

```
kbcli cluster describe-ops rose35-upgrade-6bwct -n default
```

# Migration

---

```
# Attention : only a subset versions of databases are supported
# Create a migration task to migrate the entire database under mysql: mydb1
and mytable1 under database: mydb2 to the
target mysql
kbcli migration create mytask --template apecloud-mysql2mysql
--source user:123456@127.0.0.1:3306
--sink user:123456@127.0.0.1:3305
--migration-object "'mydb1','mydb2.mytable1'"
```

Part 5

# Solution for cloud-native applications





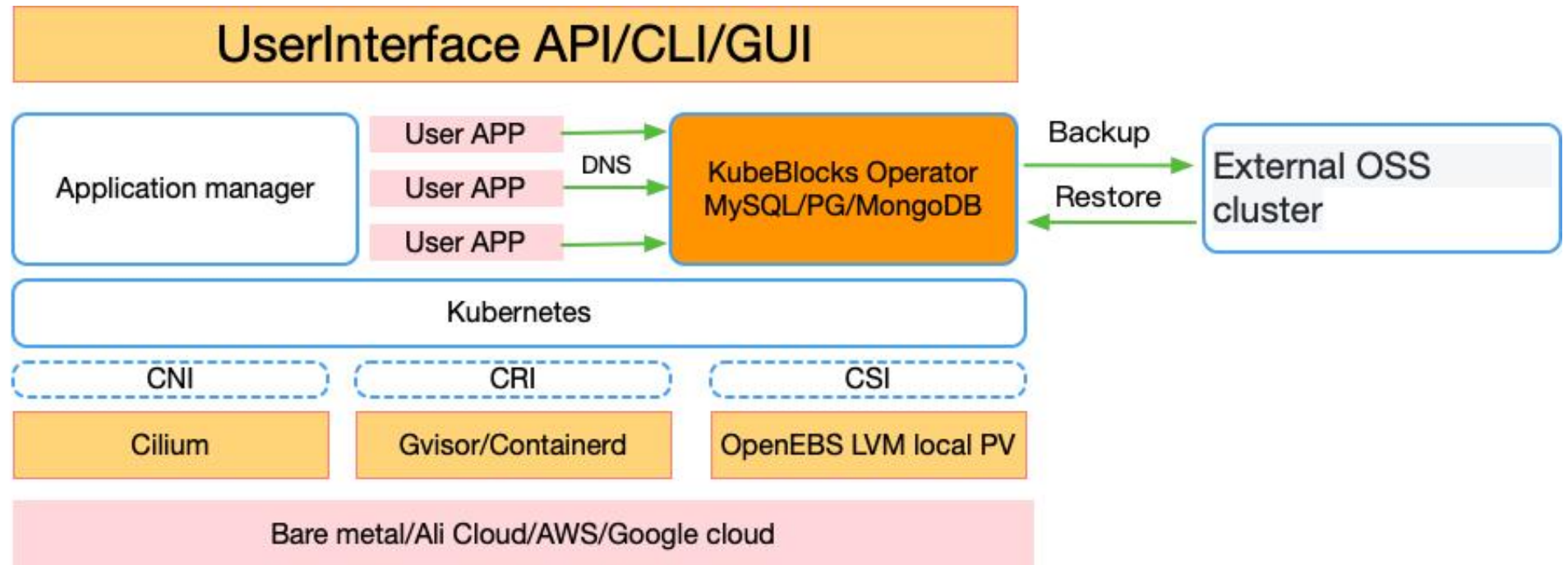
# Sealos with KubeBlocks

Sealos is a Cloud Operating System designed for managing cloud-native applications, leveraging KubeBlocks for provision of database workloads. They can be deployed in on-premises & multi-cloud environments.

All data is backed up to an OSS outside the cluster.

The user application accesses the database through DNS.

Using openEBS for providing underlying storage to the database, leveraging local disk performance and using LVM to provide tenant storage isolation.



# What about us

All great things start from scratches.

All great minds have a percentage of madness.

[kubeblocks.slack.com](https://kubeblocks.slack.com)



[kubeblocks@WeChat](https://kubeblocks@wechat.com)

