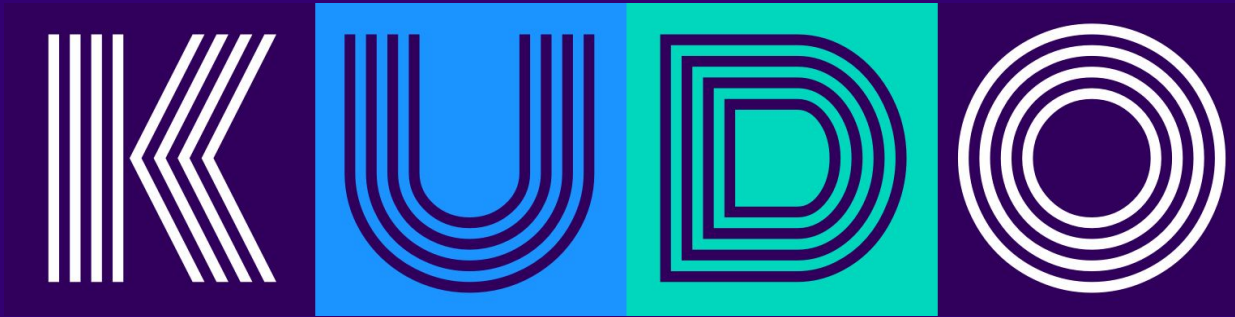

Introducing



Kubernetes Operators - The Easy Way

Nick Jones, D2iQ

\$ whoami

Nick Jones

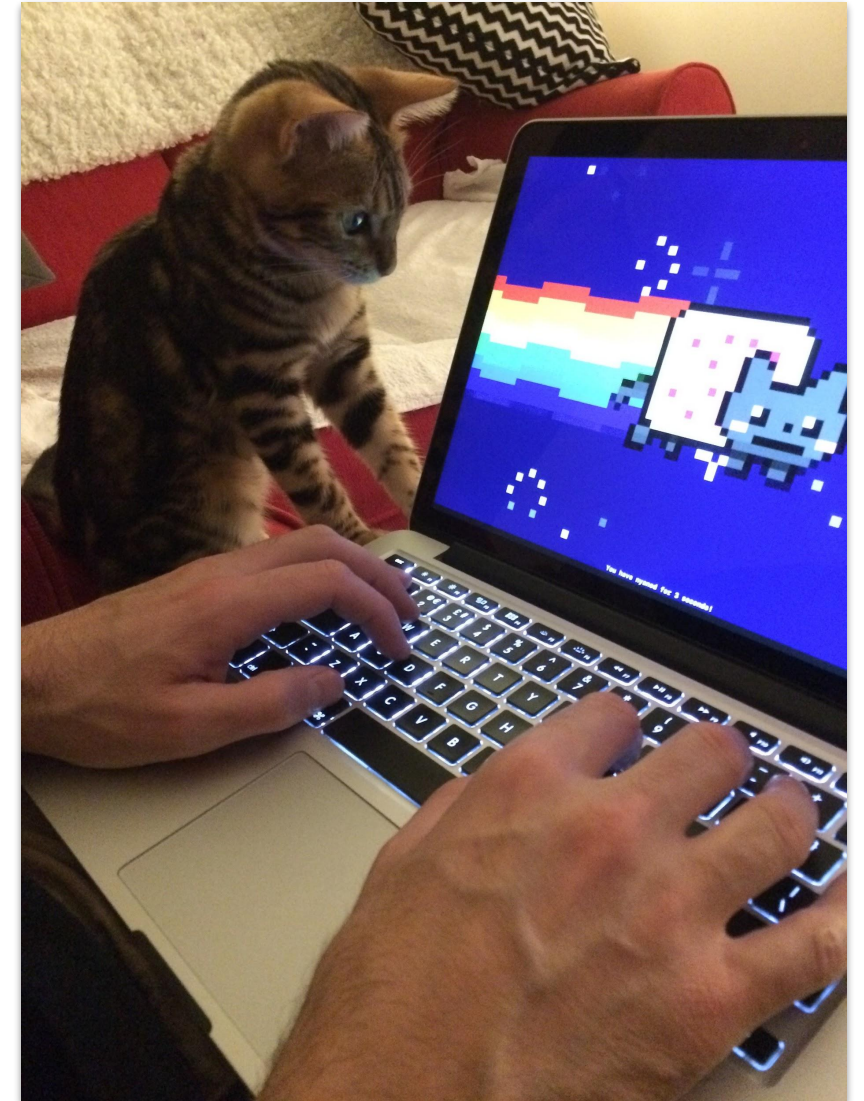
- Community Engineering Lead @ D2iQ
- Building stuff with open source software for ~20 years
- Ops, Dev and Dev/Ops
- Relatively new to Kubernetes
- ... but not new to Ops
- Likes cats




@yankcrime

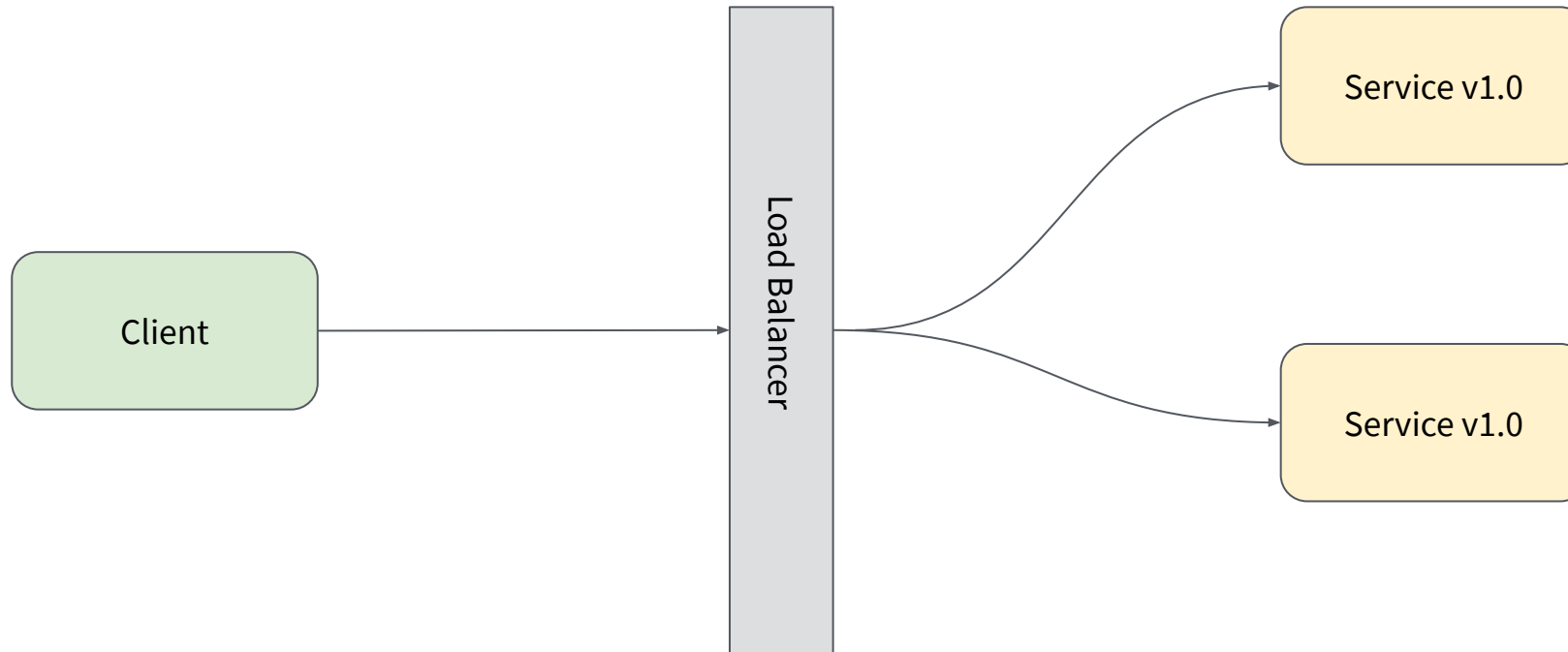


yankcrime



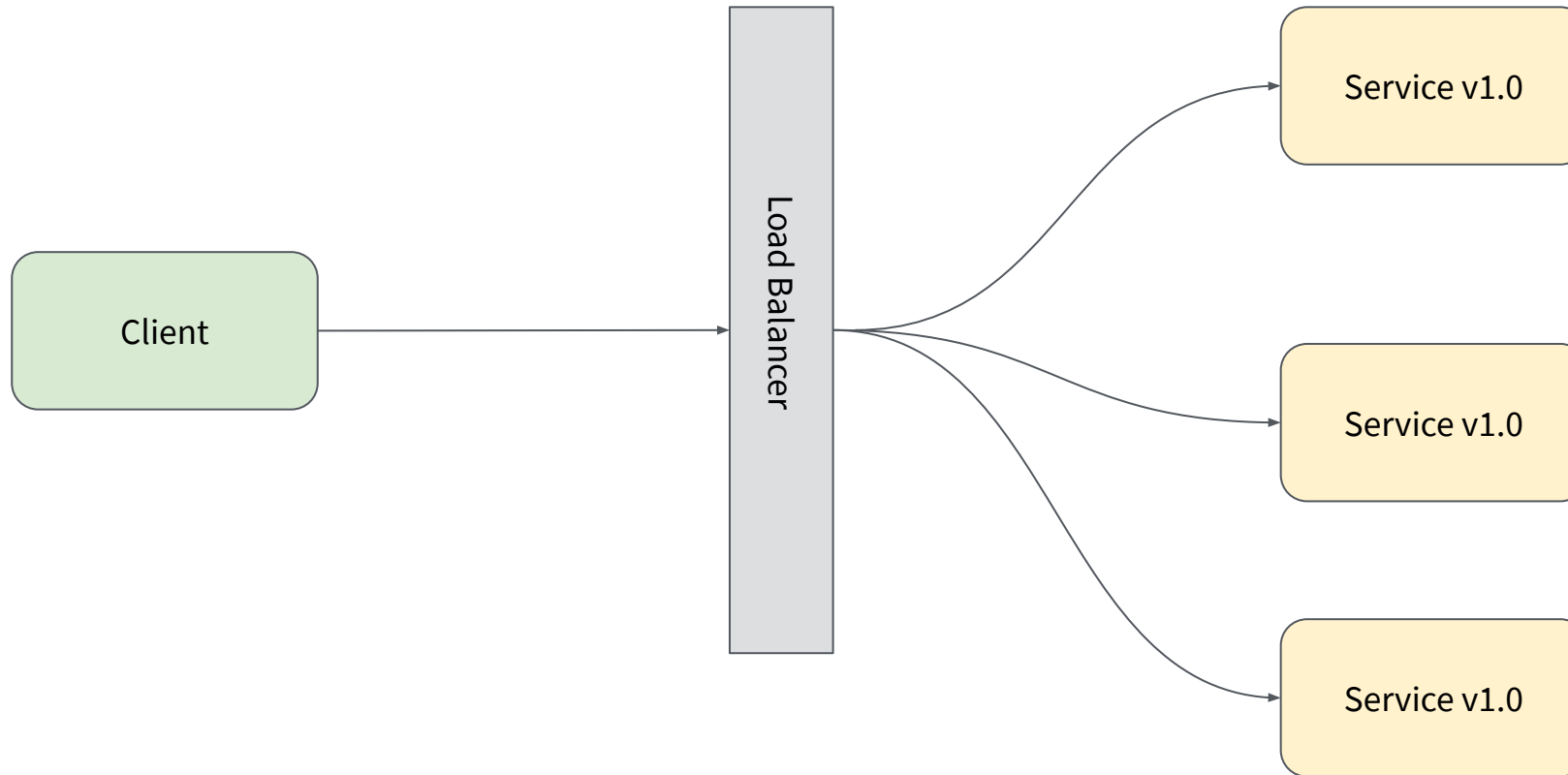
- Stateful vs. Stateless
- Kubernetes StatefulSets
- Kubernetes Operators
- 
 - Background
 - Concepts
 - Demo 🙌
 - Future
 - Getting involved

Stateless Applications



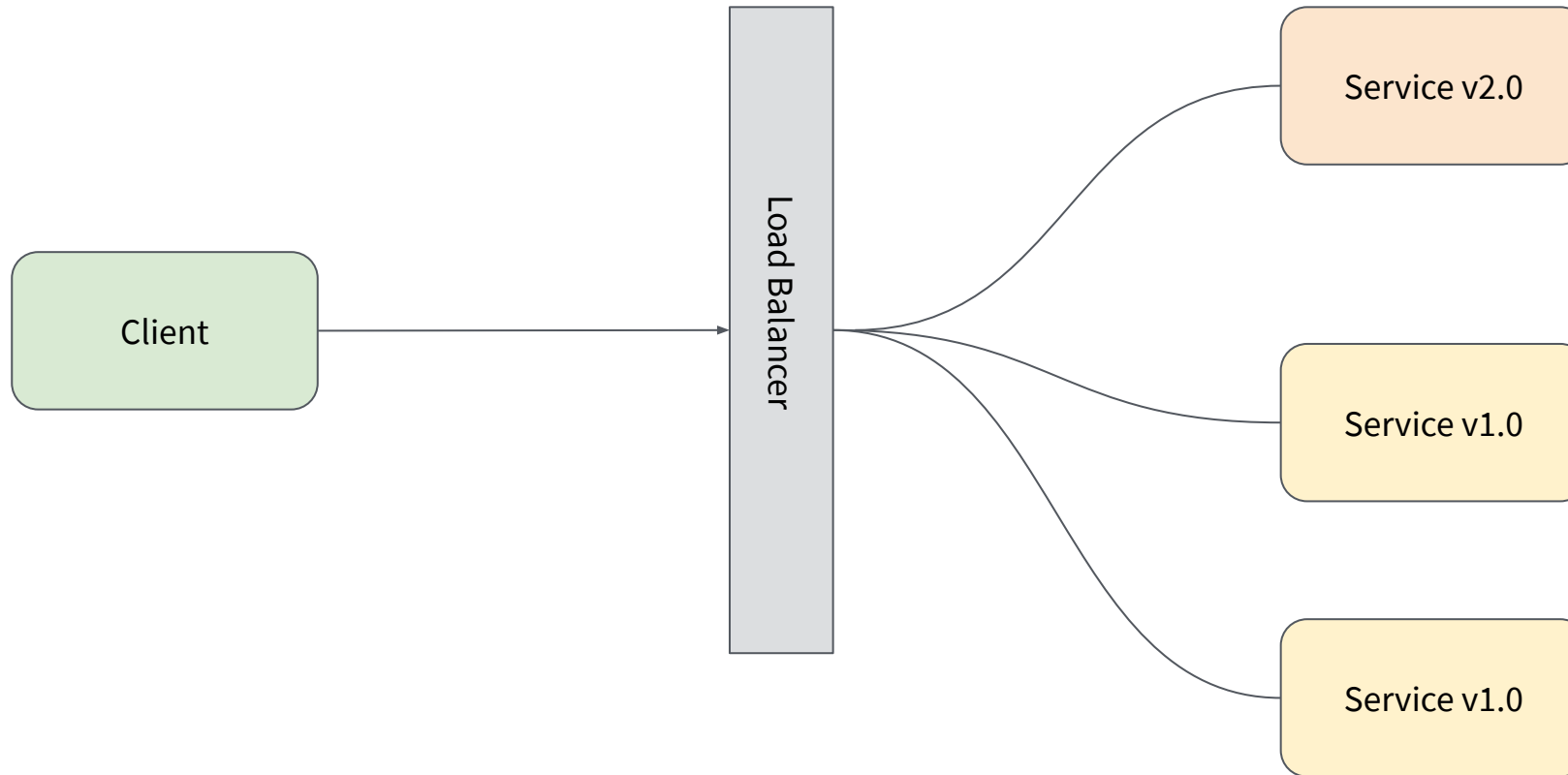
- No state persisted

Stateless Applications



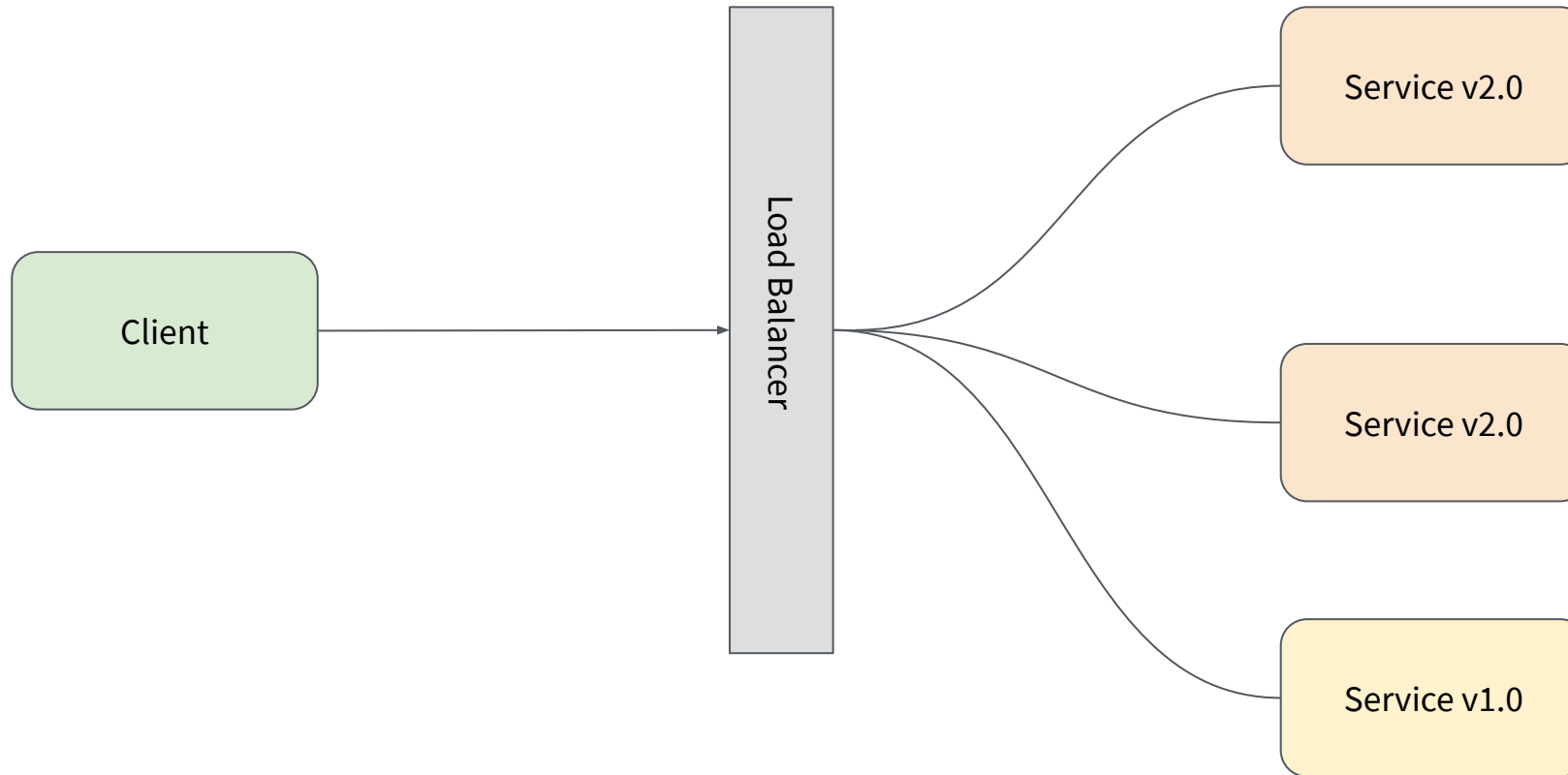
- No state persisted
- Easy to scale up / down

Stateless Applications



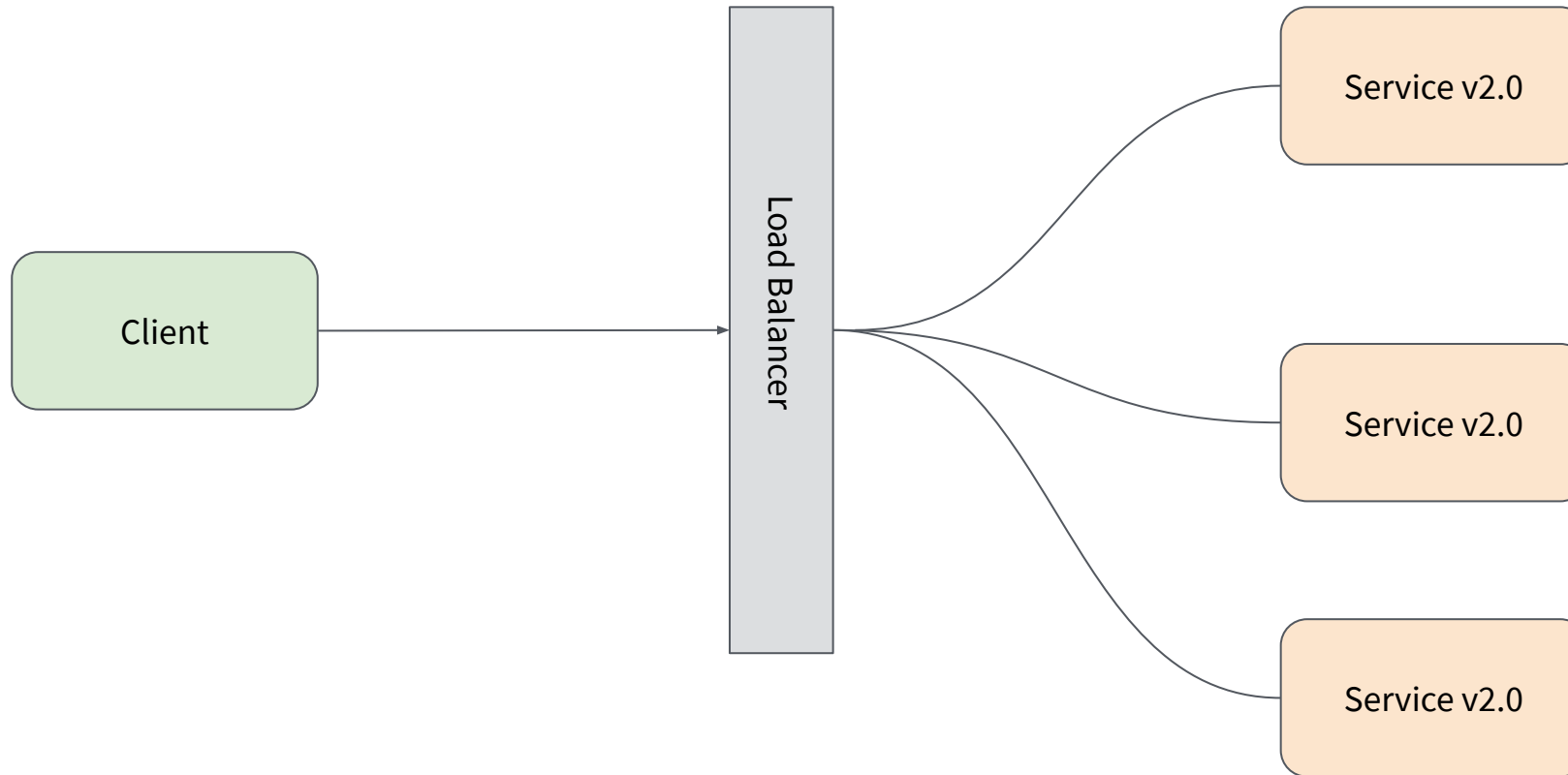
- No state persisted
- Easy to scale up / down

Stateless Applications



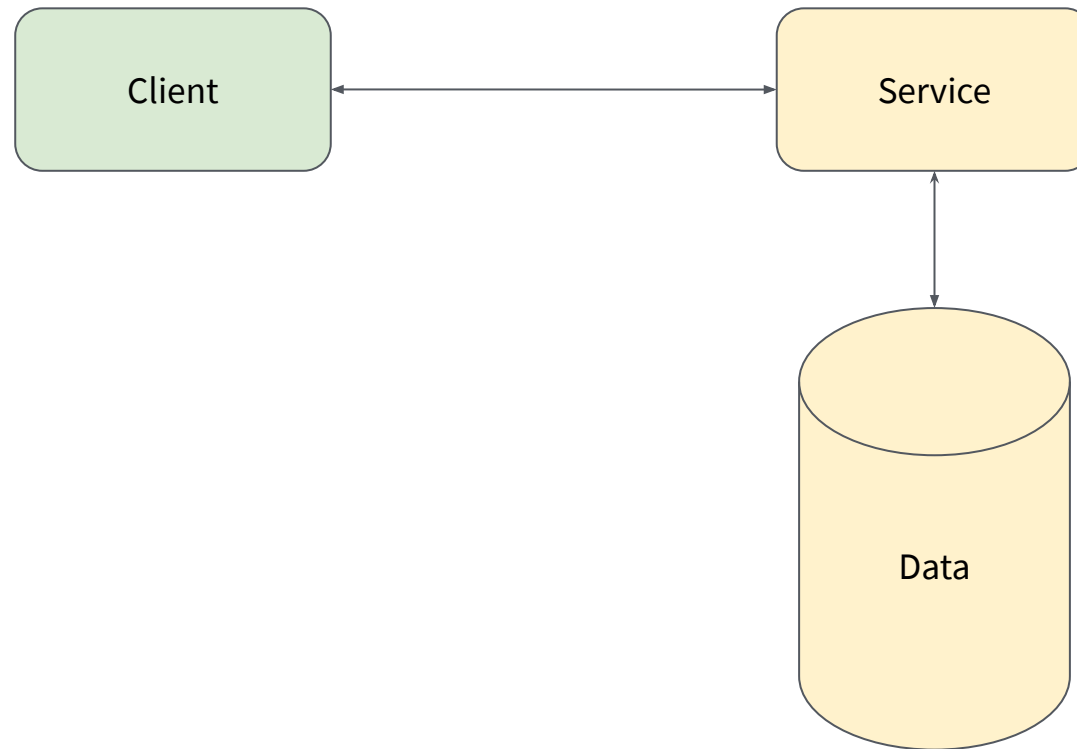
- No state persisted
- Easy to scale up / down

Stateless Applications

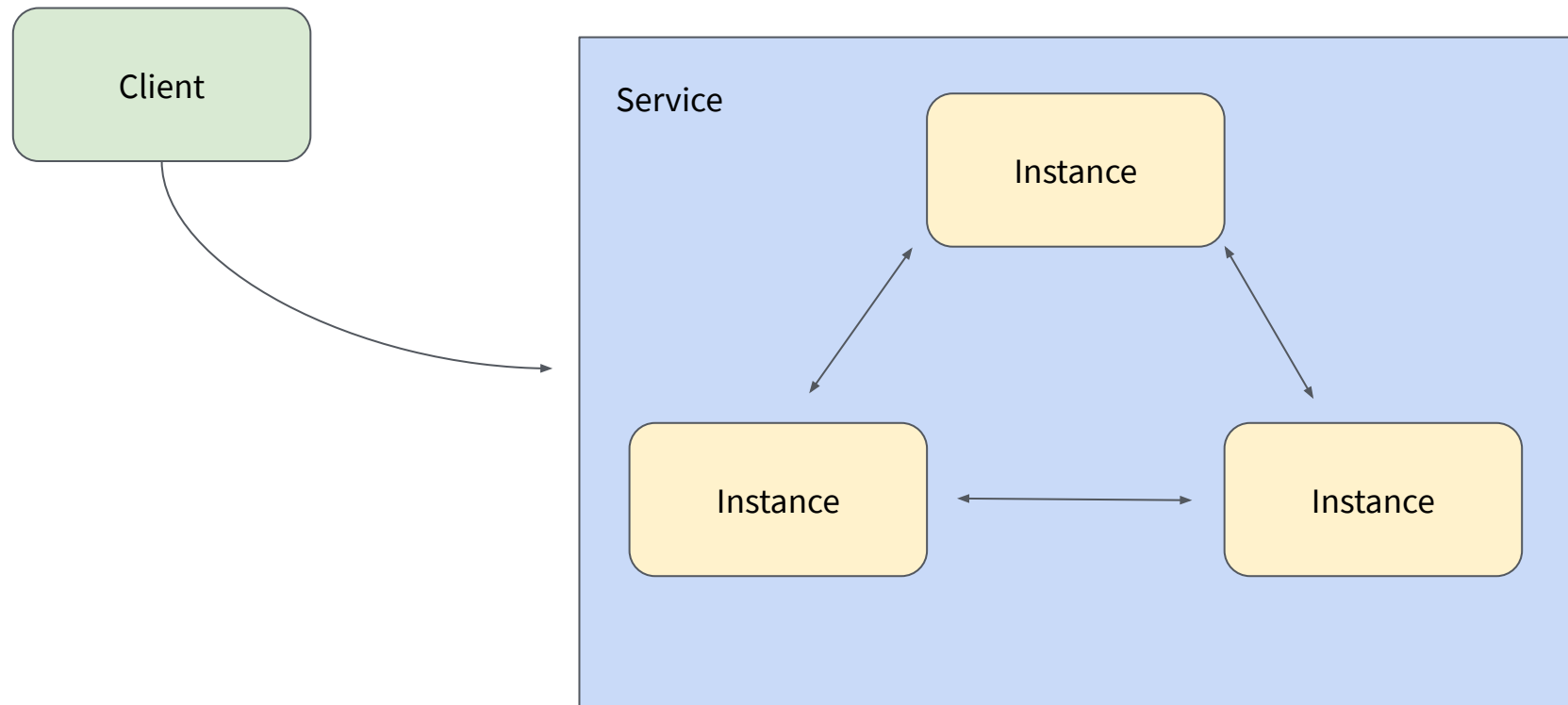


- No state persisted
- Easy to scale up/down
- Straightforward to upgrade

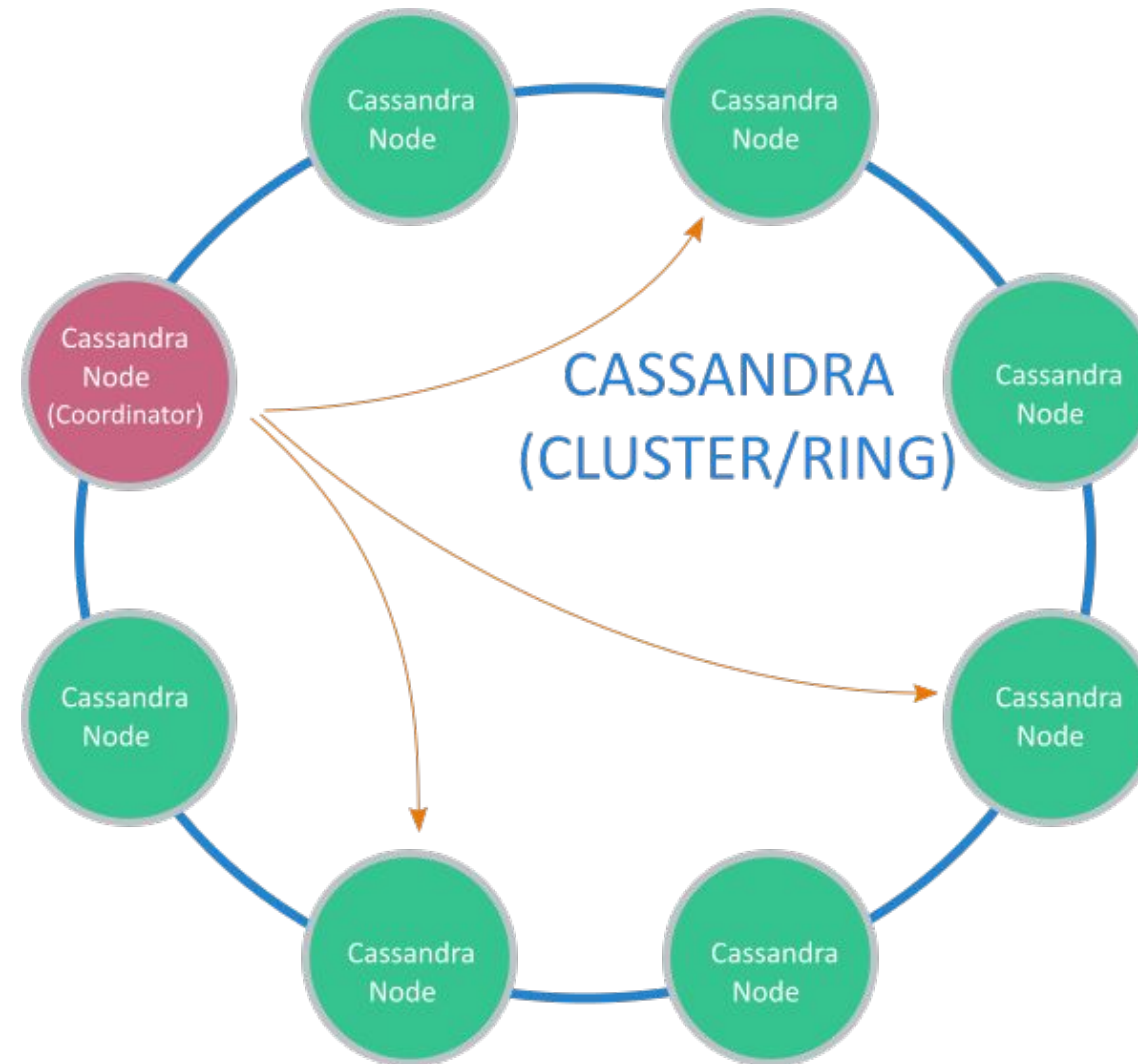
Stateful Applications



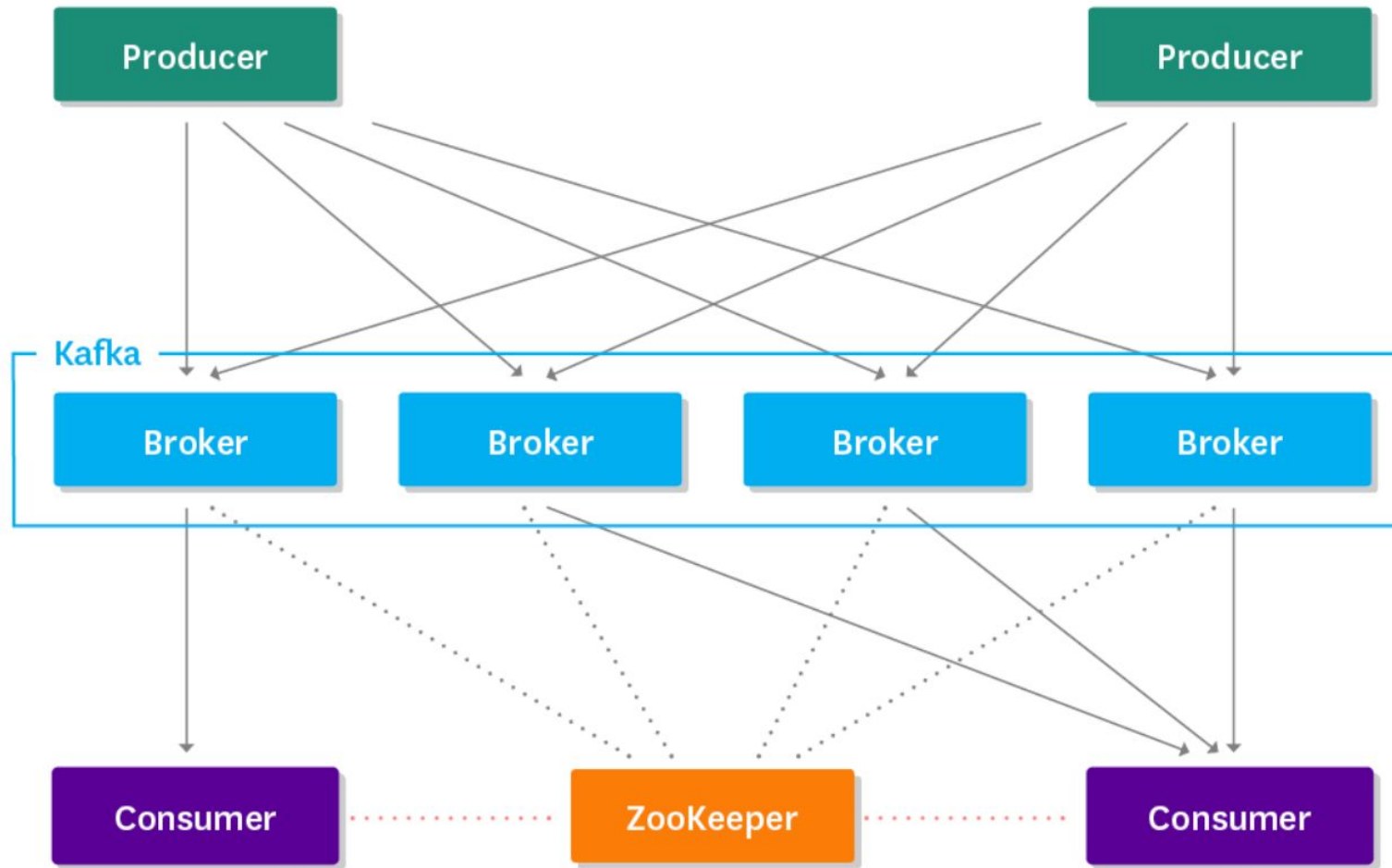
Stateful Applications



Distributed Stateful Applications



Distributed Stateful Applications





kubernetes

- Focused initially for purely stateless workloads
- Scheduler can move pods around

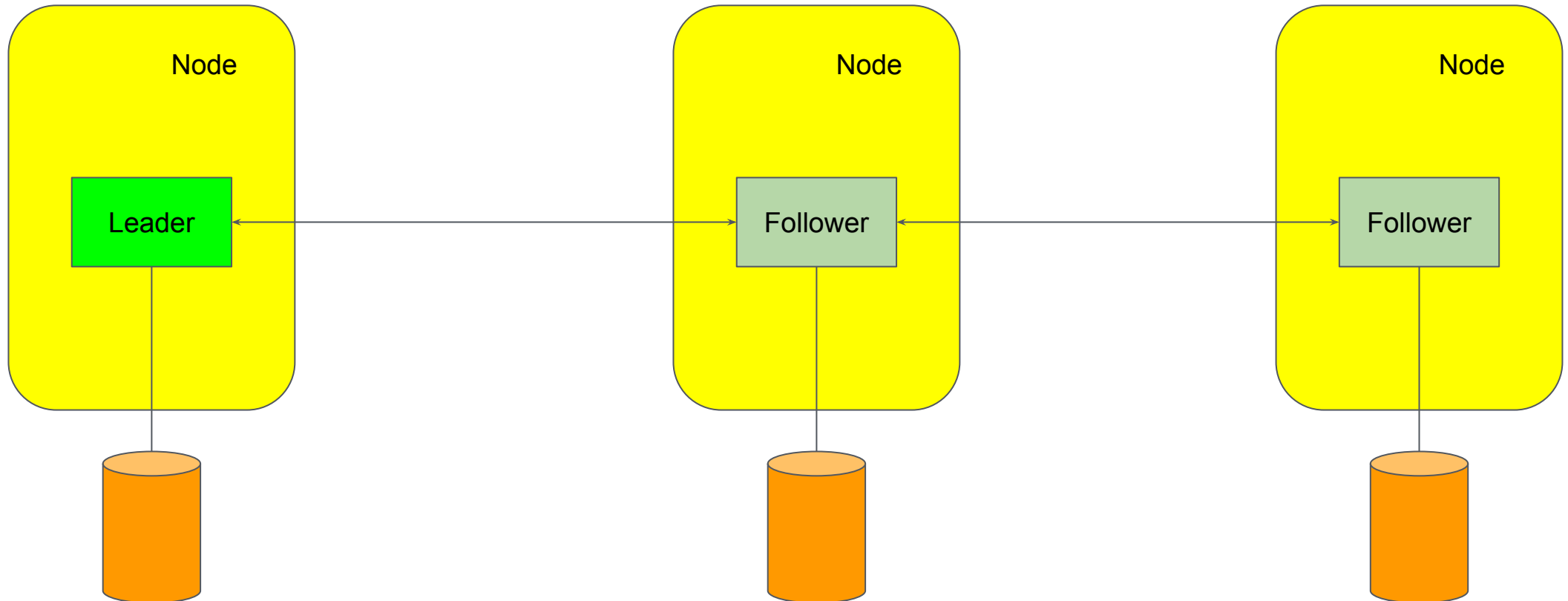


kubernetes

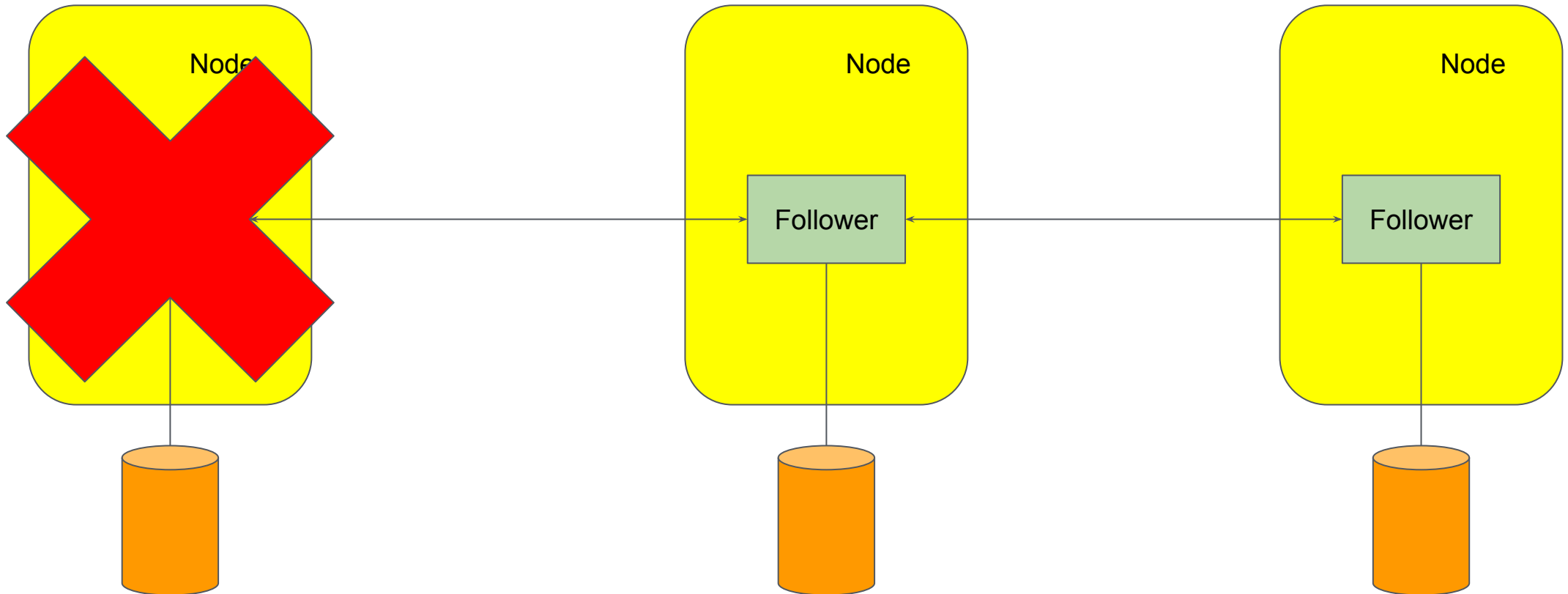
StatefulSets are valuable for applications that require one or more of the following:

- Stable, unique network identifiers.
- Stable, persistent storage.
- Ordered, graceful deployment and scaling.
- Ordered, graceful deletion and termination.
- Ordered, automated rolling updates.

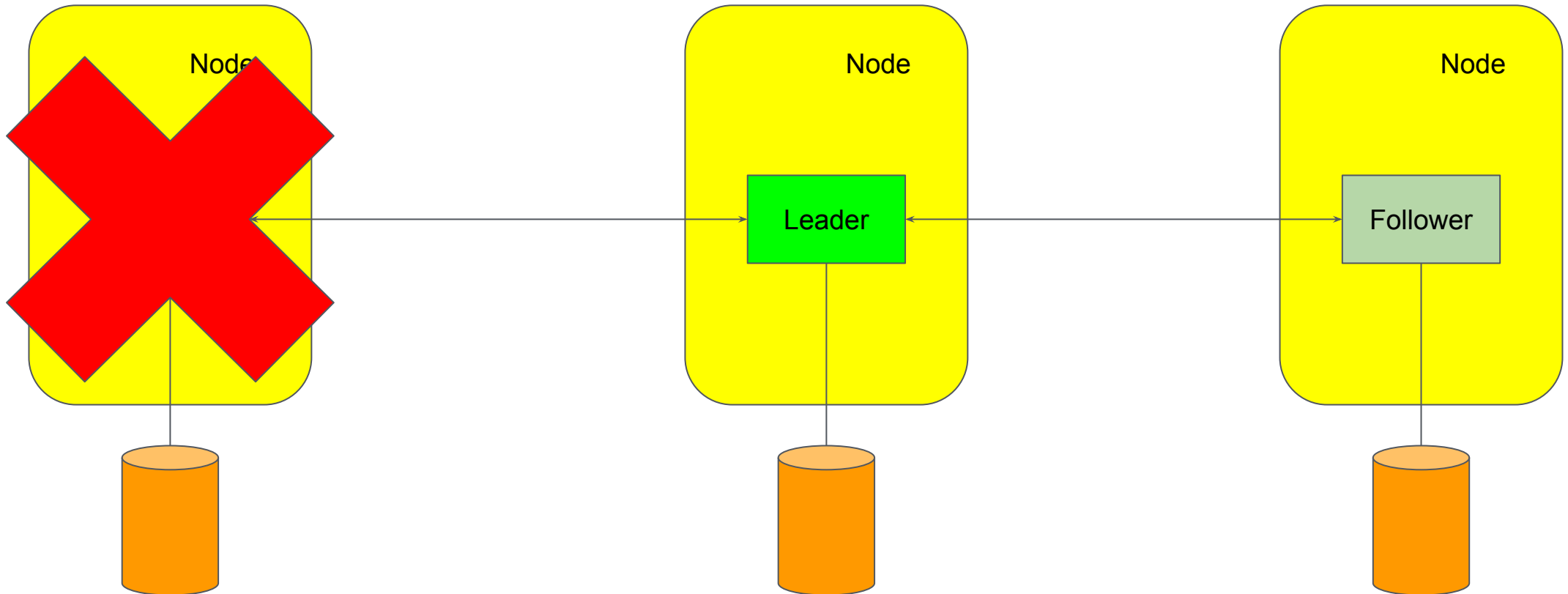
Kubernetes - StatefulSets



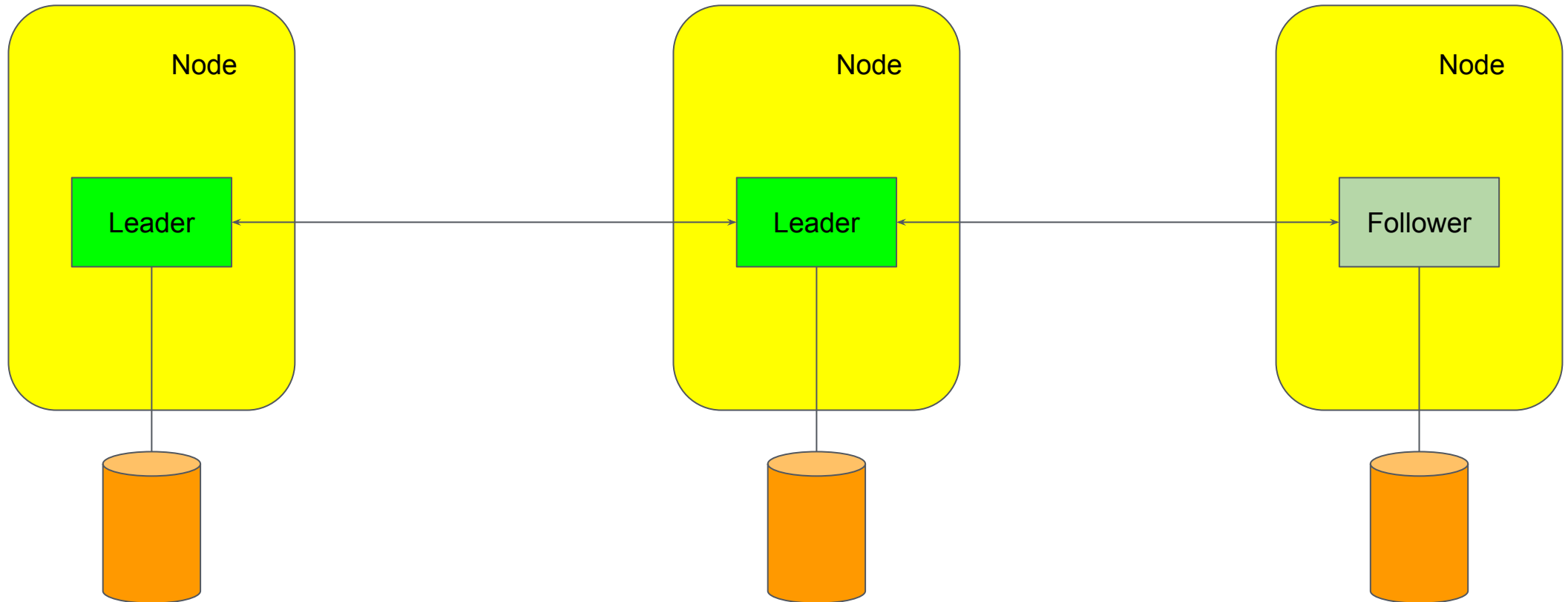
Kubernetes - StatefulSets



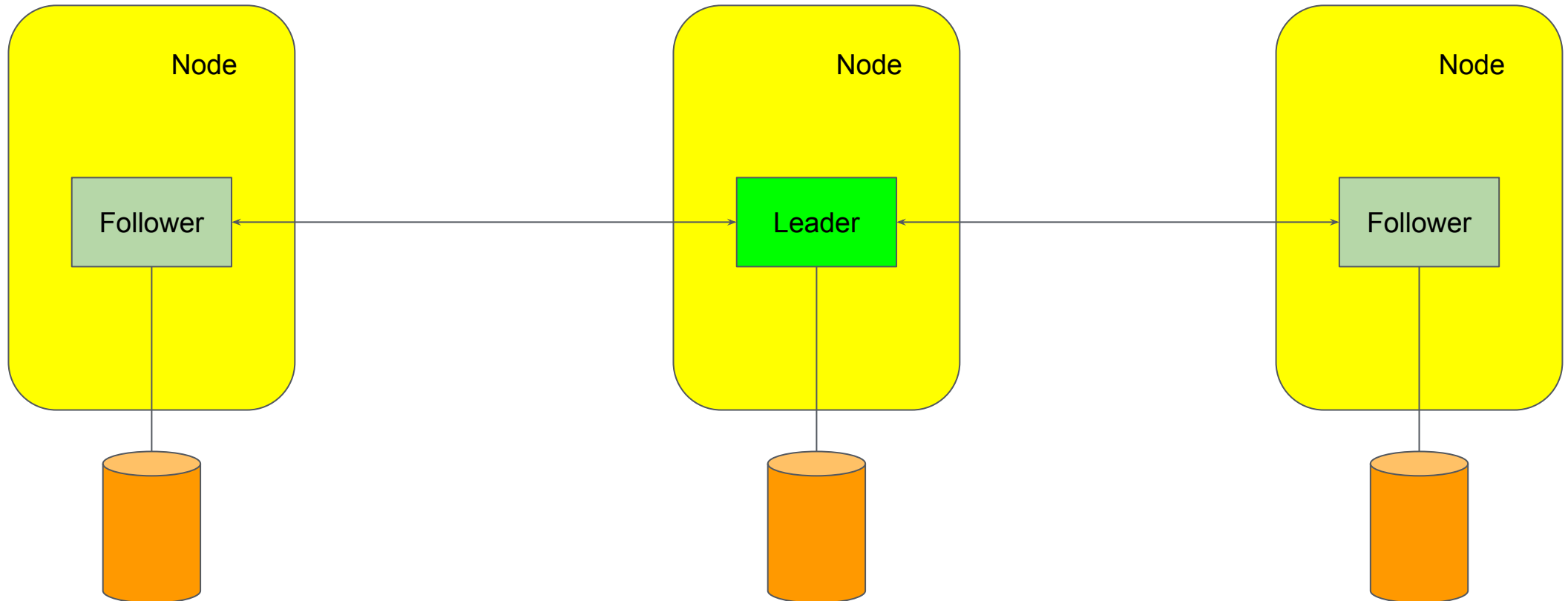
Kubernetes - StatefulSets



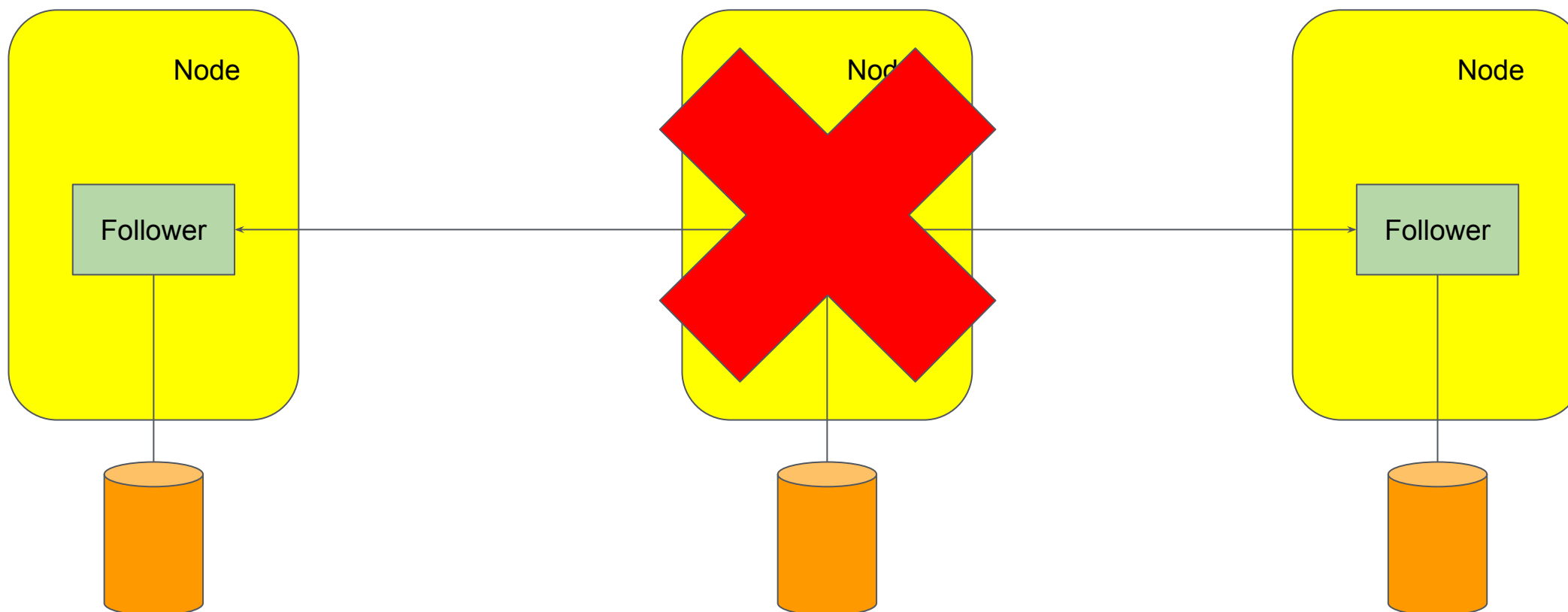
Kubernetes - StatefulSets



Kubernetes - StatefulSets



Kubernetes - StatefulSets





kubernetes



Kelsey Hightower ✓

@kelseyhightower

Following



I'm always going to recommend people exercise extreme caution when running stateful workloads on Kubernetes. Most people who are asking "can I run stateful workloads on Kubernetes" don't have much experience with Kubernetes and often times the workload they are asking about.

3:10 AM - 24 Mar 2019

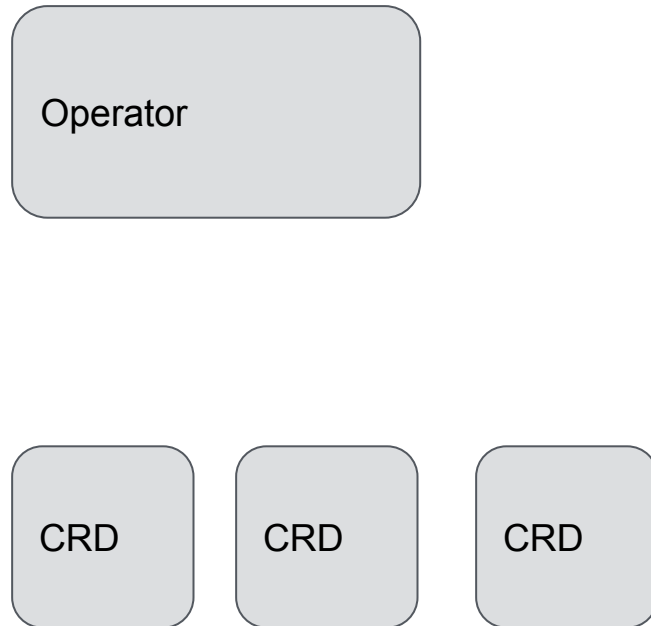
286 Retweets 901 Likes



Kubernetes Operators



- Orchestrate stateful applications using K8s API
- Extend API using Custom Resource Definitions
- Encode domain specific operational knowledge
- Upgrades
- Failure and Recovery Scenarios
- Scaling up / down
- Purpose built per application
- “Kubernetes is an Operations API”:
 - <https://blog.atomicinc.com/2018/05/23/kubernetes-is-an-operations-api/>



- Operator manages and monitors lifecycle
- CRD's represent application elements / actions

```
apiVersion:
mysql.presslabs.org/v1alpha1
kind: MysqlCluster
metadata:
  name: my-cluster
spec:
  replicas: 2
  secretName: my-secret
```

```
$ kubectl apply -f mysql-cluster.yaml
```

Operator Framework

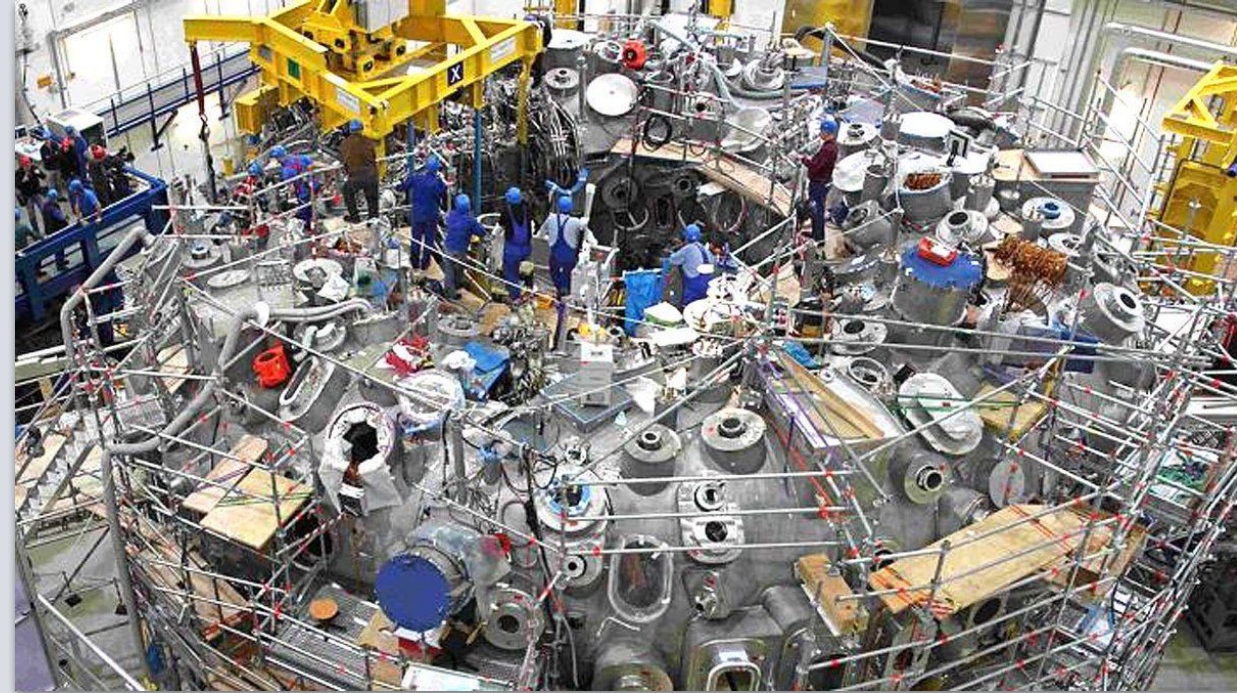
- RedHat / IBM project
- Implement using Ansible, Helm charts, or Go
- Existing implementations often don't cover the entire lifecycle
- Ansible and Helm are limited. Go requires 1,000s of lines of controller code

Kubebuilder

- Kubernetes SIG API Machinery sub-project
- Operators written in Go with a focus on code generation
- Existing implementations often don't cover the entire lifecycle

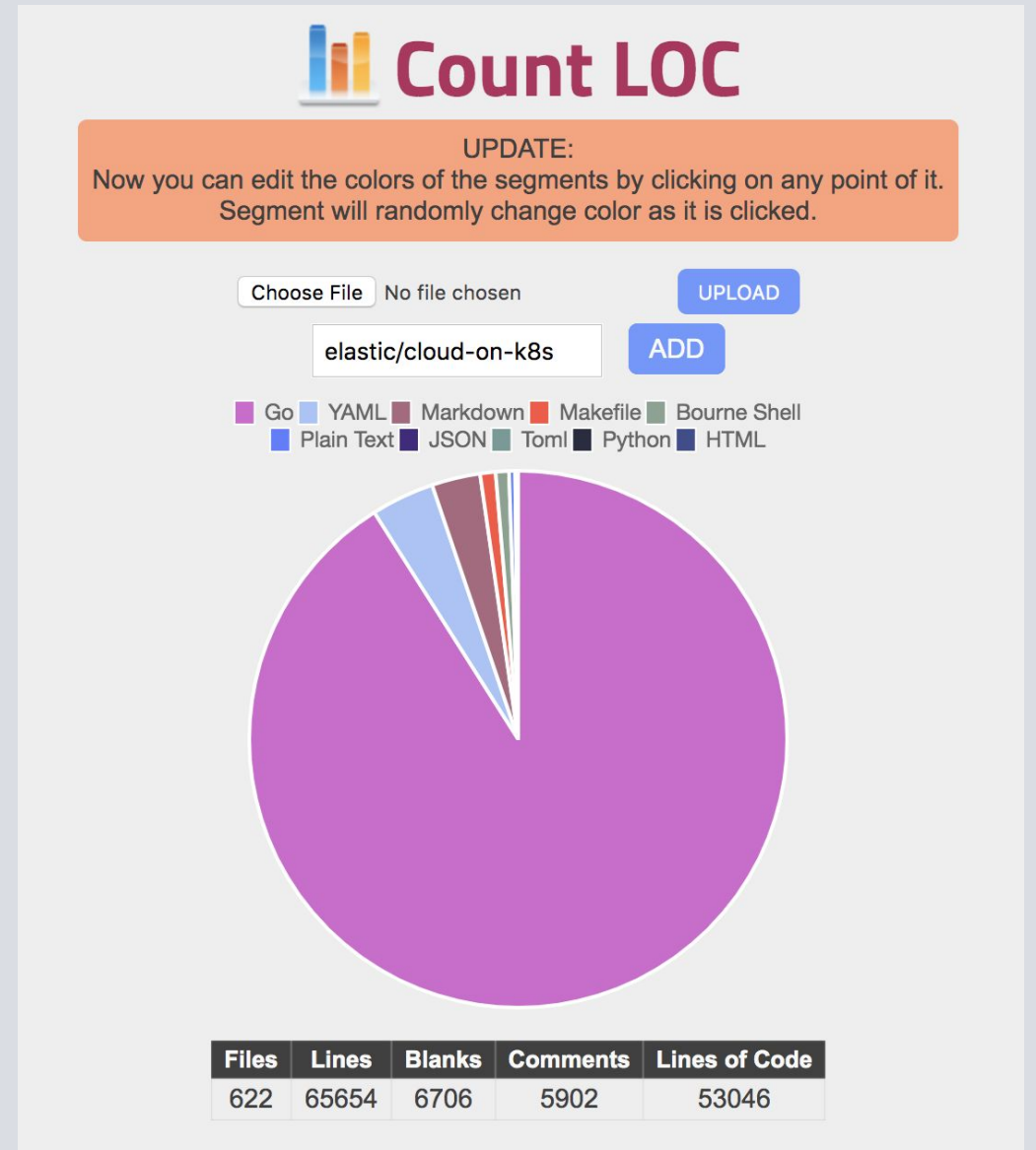
Developing Operators

- Operators require deep knowledge of Kubernetes internals
- Significant software development undertaking
- May require (10s of) thousands of lines of code
- Controller sprawl can be a thing



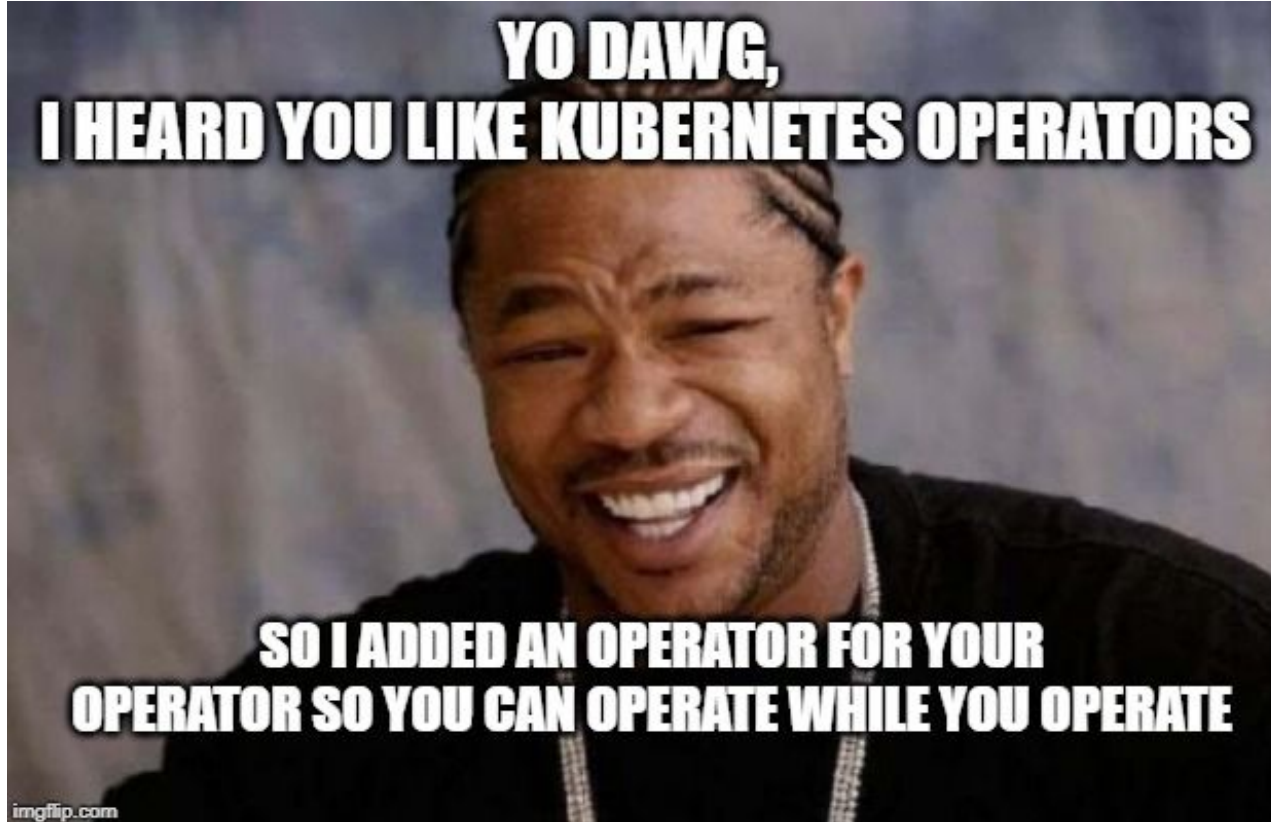
Developing Operators

- Operators require deep knowledge of Kubernetes internals
- Significant software development undertaking
- May require (10s of) thousands of lines of code
- Controller sprawl can be a thing



👋 operators 👋 operators 👋 operators 👋 operators

operators



Lachlan Evenson @LachlanEvenson · 4d

OH: You're going to need an operator to operate that operator

9

2

65



Andrew Block @sabre1041 · 4d

and behold the meta-operator

2

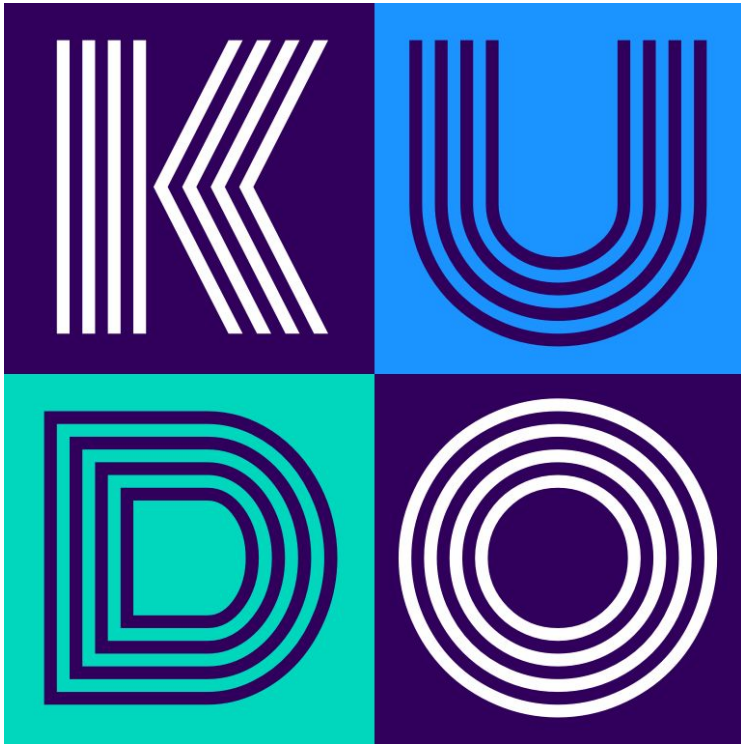
4



Gaunerd
@gaunetes

Replying to @sabre1041 and @LachlanEvenson

Aka KUDO



- Kubernetes Universal Declarative Operator
- A toolkit and runtime for building operators
- Encodes commonality and reuse between lifecycle operations
- Optimised for complex, stateful applications
- Increases developer productivity when **building operators**
- Increases operator productivity when **operating services**
- OS project licensed as Apache 2.0

Operator Framework

- RedHat / IBM project
- Implement using Ansible, Helm charts, or Go
- Existing implementations often don't cover the entire lifecycle
- Ansible and Helm are limited. Go requires 1,000s of lines of controller code

Kubebuilder

- Kubernetes SIG API Machinery sub-project
- Operators written in Go with a focus on code generation
- Existing implementations often don't cover the entire lifecycle

KUDO

- Polymorphic
- Universal Operator
- Built using community projects (Kubebuilder, Kustomize, ...)
- Write Operators as templated YAML manifests
- Provide high level CRDs that represent workloads
- Focused on higher level coordination of software lifecycles
- “Day 2 Operators”

How KUDO Helps Developers



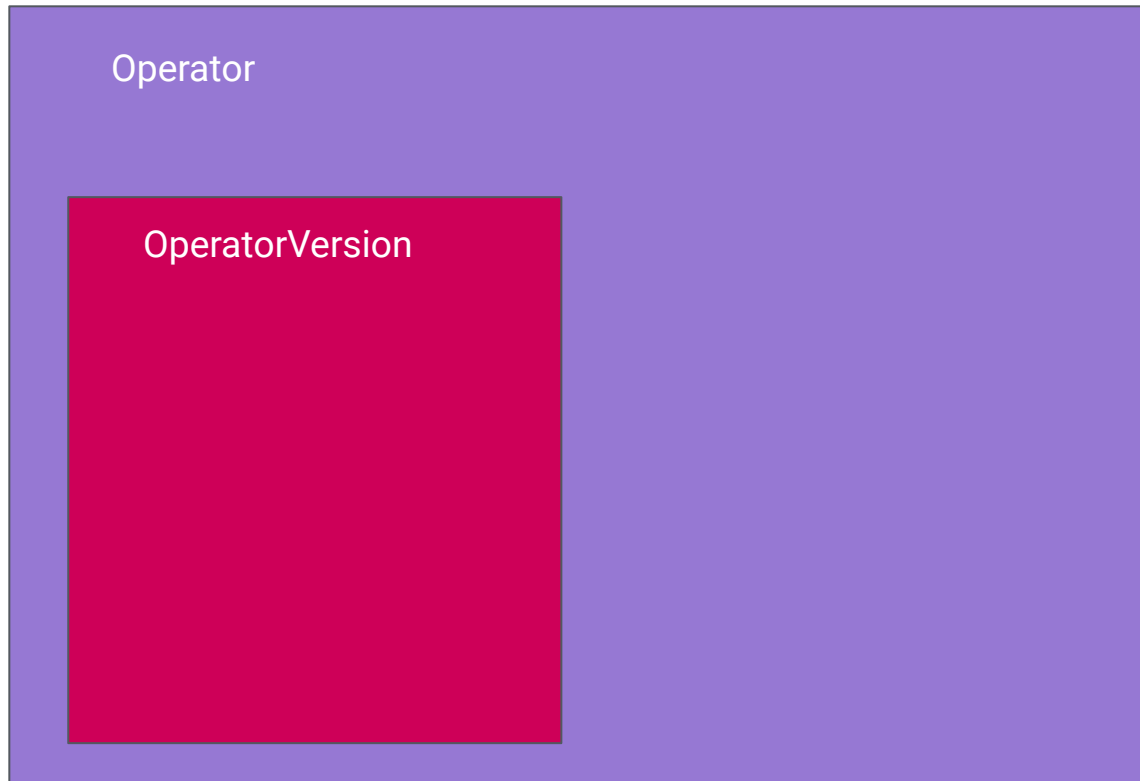
- Provides abstractions for sequencing lifecycle operations using Kubernetes objects and “plans”, conceptually similar to runbooks
- Encodes commonality and reuse between lifecycle operations
- Reduces boilerplate and code duplication between Operators
- Provides an extension mechanism to create “flavors” of a base Operator for customisation specific to a user’s environment
- Provides ISVs with a tool to ship best practices for Day 2 operations alongside their software
- Ships with testing tool to enable TDD of Kubernetes resources

- KUDO provides the `kubectl kudo` plugin to deploy, manage and debug their workloads
 - It's possible to just use kubectl - KUDO is Kubernetes!
- As it's common to deploy multiple Operators to a cluster, KUDO provides a similar API and CLI / workflow experience for all
- All workloads are managed as CRDs, facilitating GitOps
- Existing Operators can be managed by KUDO, natively understanding how to deploy CRDs, custom resource, and other operators, enabling dependencies as part of other workloads
- (Future) Centralised supportability, metrics / alerting, as well as security and RBAC features for Enterprise workloads

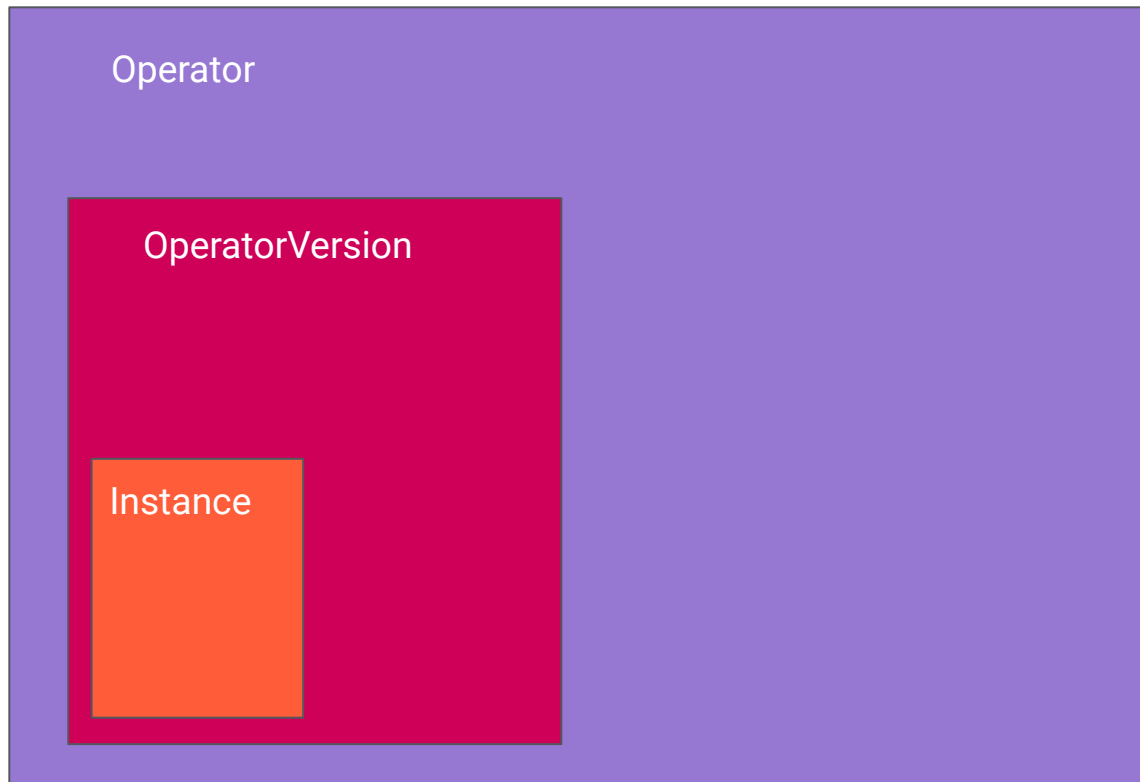
A large purple rectangle representing the 'Operator' concept. The word 'Operator' is written in white text in the top-left corner of the rectangle.

Operator

- High level description of a deployable service
- A deployable service can be anything that you'd want to run on your cluster
- Represented as a CRD object

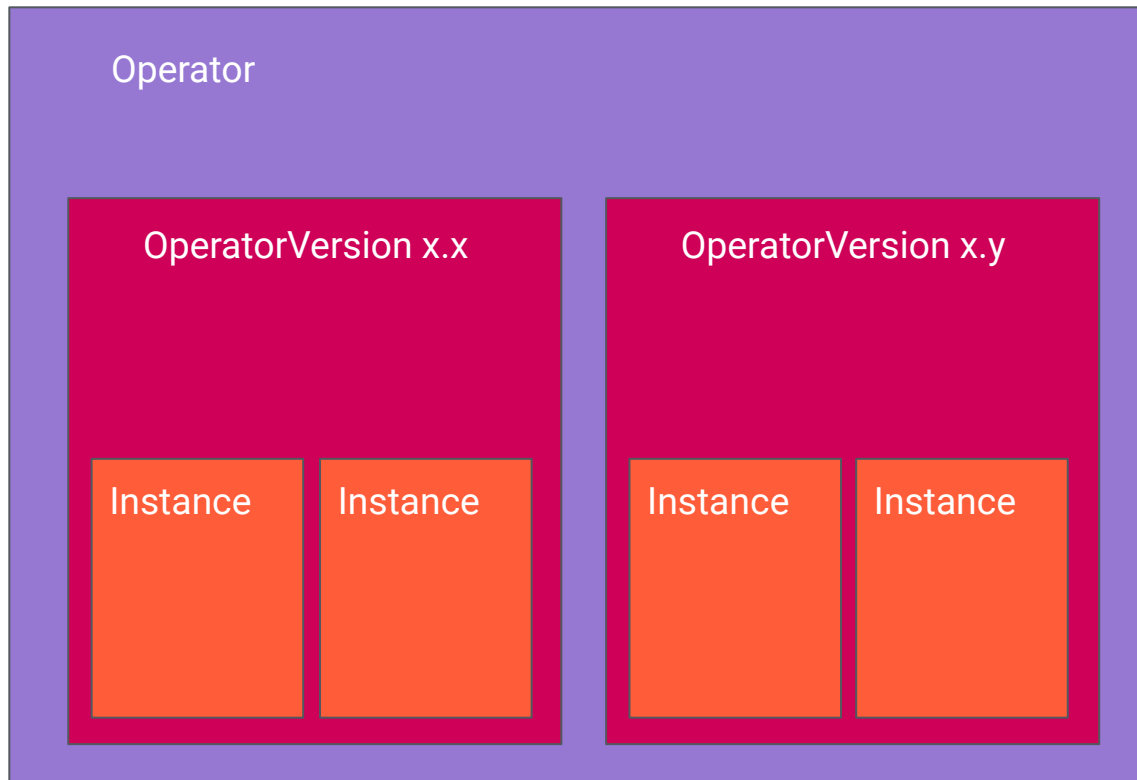


- Implementation of an Operator
- Specific version of a deployable application
- Contains parameters, objects, plans



- Ties application instantiation to an OperatorVersion
- Once created, renders parameters in templates such as services, pods or StatefulSets
- Can create multiple instances of an OperatorVersion within your cluster

KUDO Concepts - *Instance*



- Ties application instantiation to an OperatorVersion
- Once created, renders parameters in templates such as services, pods or StatefulSets
- Can create multiple instances of an OperatorVersion within your cluster

```
Plan foo
├─ Phase bar
│   ├── Step qux
│   └─ Step quux
└─ Phase baz
    ├── Step quuz
    ├── Step corge
    └─ Step grault
```

- Orchestrate tasks through phases and steps
- A structured 'runbook' which can then be executed by software
- Typically define several plans:
 - Deploy
 - Backup
 - Restore
 - Upgrade
- Phases and steps can be run serial or parallel

- CLI extension to kubectl
- Can still use 'vanilla' kubectl

```
# Install a KUDO package from the official GitHub repo.  
kubectl kudo install <name> [flags]
```

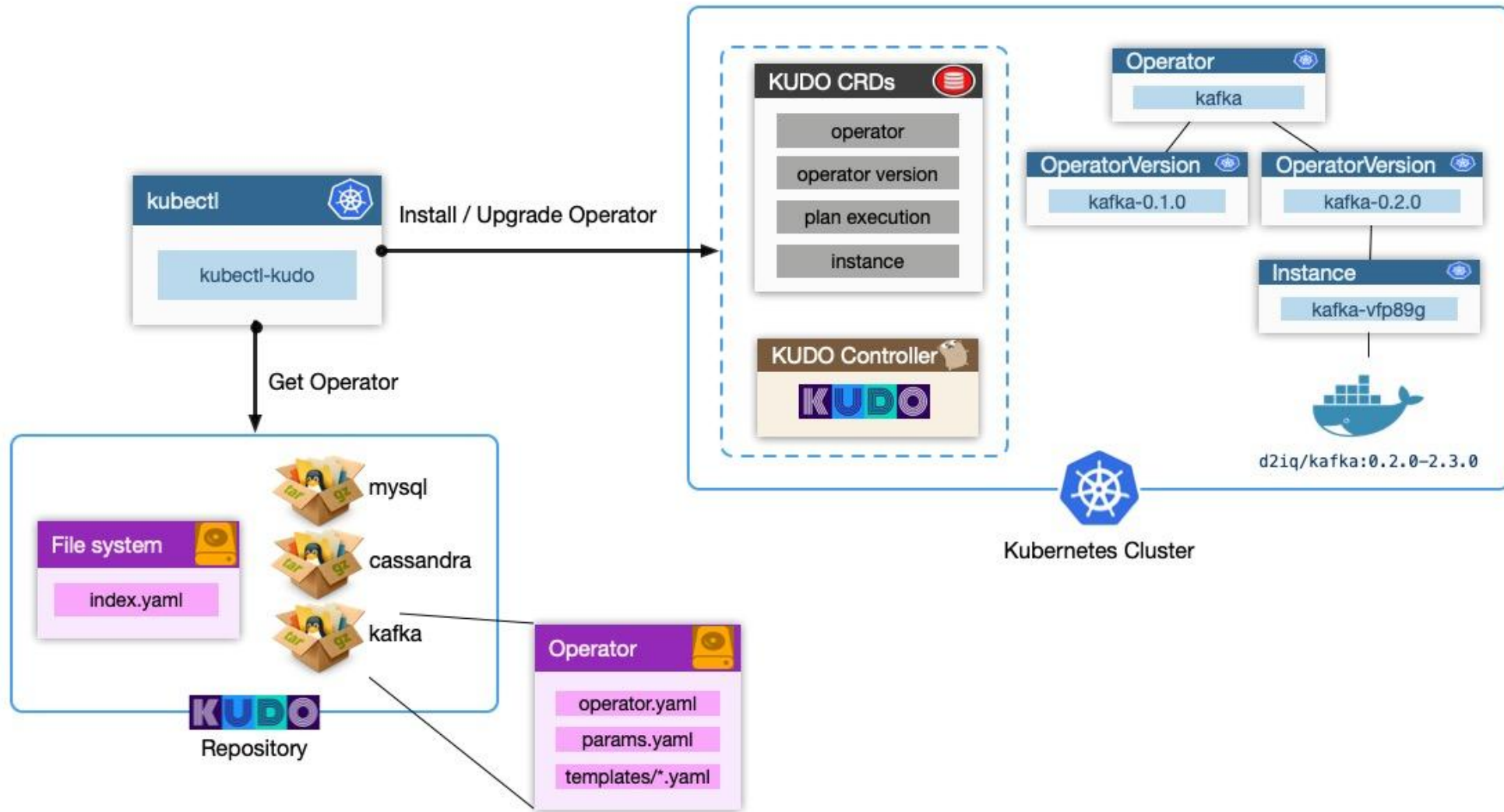
```
# View plan history of a specific package  
kubectl kudo plan history <name> [flags]
```

```
# View all plan history of a specific package  
kubectl kudo plan history [flags]
```

```
# List instances  
kubectl kudo list instances [flags]
```

```
# View plan status  
kubectl kudo plan status [flags]
```

KUDO Concepts - *Architecture*



Demo - <https://butt.holdings>

D2
IQ





- Dynamic CRDs
 - Manage the lifecycle of operator CRDs for the operator developers and users
- Operator Dependencies
 - Ability for KUDO to support a wide range of dependencies (from existing instances and connection strings to entirely new dependencies that are KUDO managed), and for tighter control of dependency specification by operator developers.
- Operator Extensions
 - Extend from other formats such as other KUDO operators, Helm charts, or CNAB bundles without forking an operator.
- Something other than YAML! Starlark or CUE likely candidates.
- Pipe Tasks
 - Generation of content which can then be 'piped' to another task
 - E.g certificate generation / creation as part of bootstrap
 - Just landed (<https://github.com/kudobuilder/kudo/pull/1105>) 🎉

- Helm chart
 - Import and extend
- Operator Development
 - Skeleton Generator
 - Also just landed 🎉
(<https://github.com/kudobuilder/kudo/pull/1206> and <https://github.com/kudobuilder/kudo/pull/1222>)
 - Linter
 - Snippet / extension library
- KUDO API
- Roadmap here: <https://github.com/orgs/kudobuilder/projects/2>

KUDO Roadmap - Operator Extensions



Operator Developer Maintained

ACME Corp Maintained

MySQL

MySQL + GKE

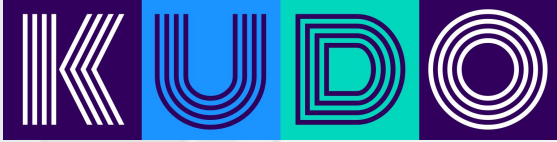
ACME Corp

“Standard” infrastructure, plans, CRDs, etc.

Istio, Cloud Storage, GCP Security Rules,
StackDriver Monitoring, etc.

ACME specific plans. Network policy, special
operations, cached queries, custom functions,
etc.

KUDO Community - Get Involved!



<https://kudo.dev/>



<https://github.com/kudobuilder/kudo>



#kudo <http://slack.k8s.io/>



<https://groups.google.com/forum/#!forum/kudobuilder>



Community Meeting - bi-weekly, Thursdays 10am PT