Introducing KUDO

Kubernetes Operators The Easy Way

MESOSPHERE

2019

Matt Jarvis, Senior Director - Open Source Program Office

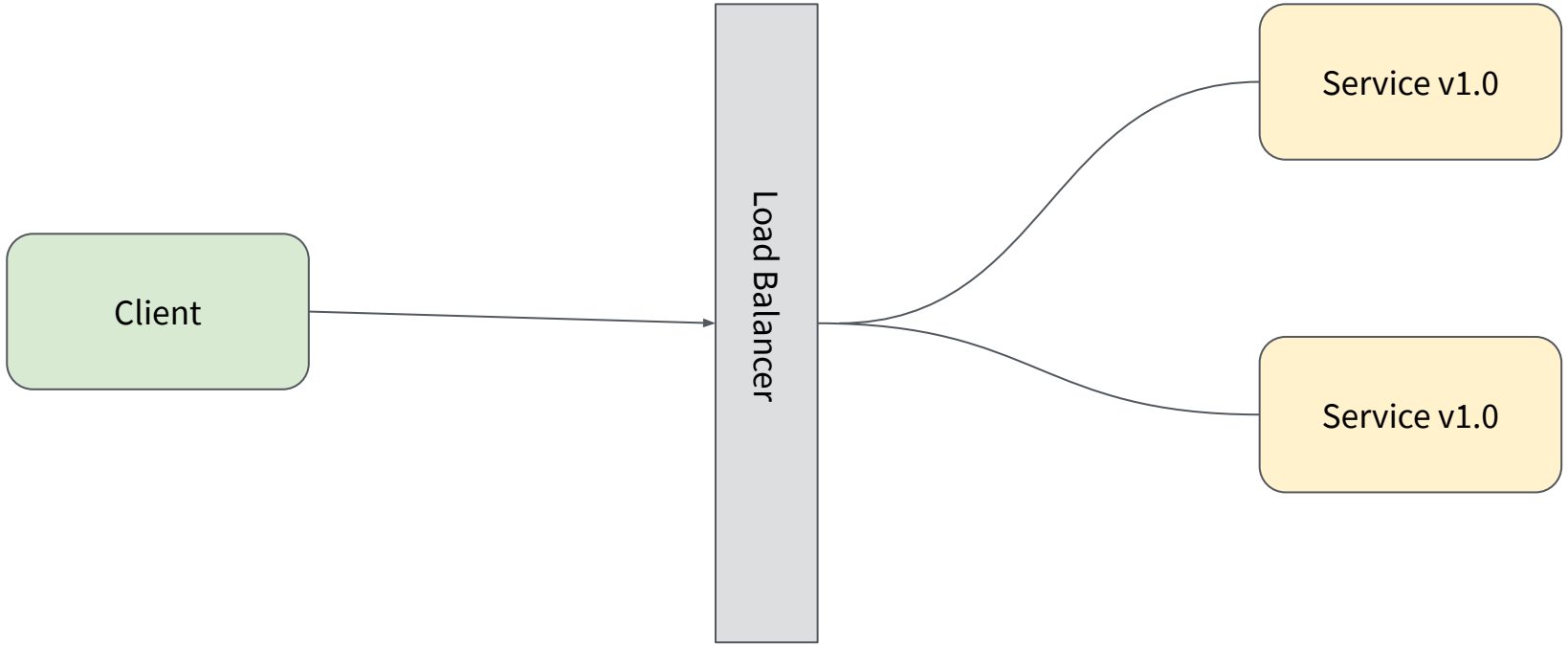
- Open Source Program Office @ Mesosphere
- Building stuff with open source software for ~20 years
- Ops, Dev and Dev/Ops
- Relatively new to Kubernetes
- … but not new to Ops

@mattj_io

Matt Jarvis, Senior Director - Open Source Program Office

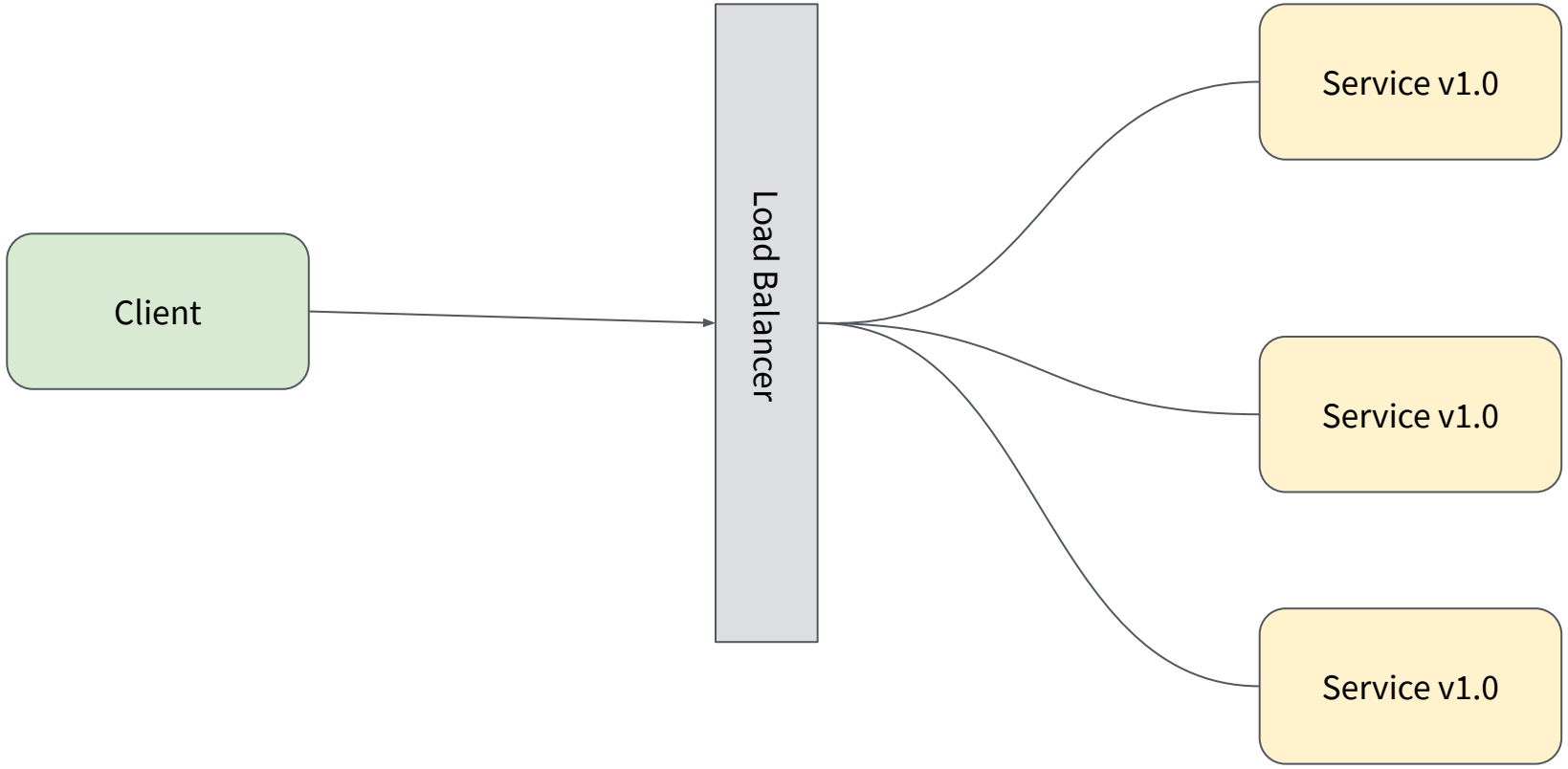
- Open Source Program Office @ Mesosphere
- Building stuff with open source software for ~20 years
- Ops, Dev and Dev/Ops
- Relatively new to Kubernetes
- … but not new to Ops

Stateless Applications



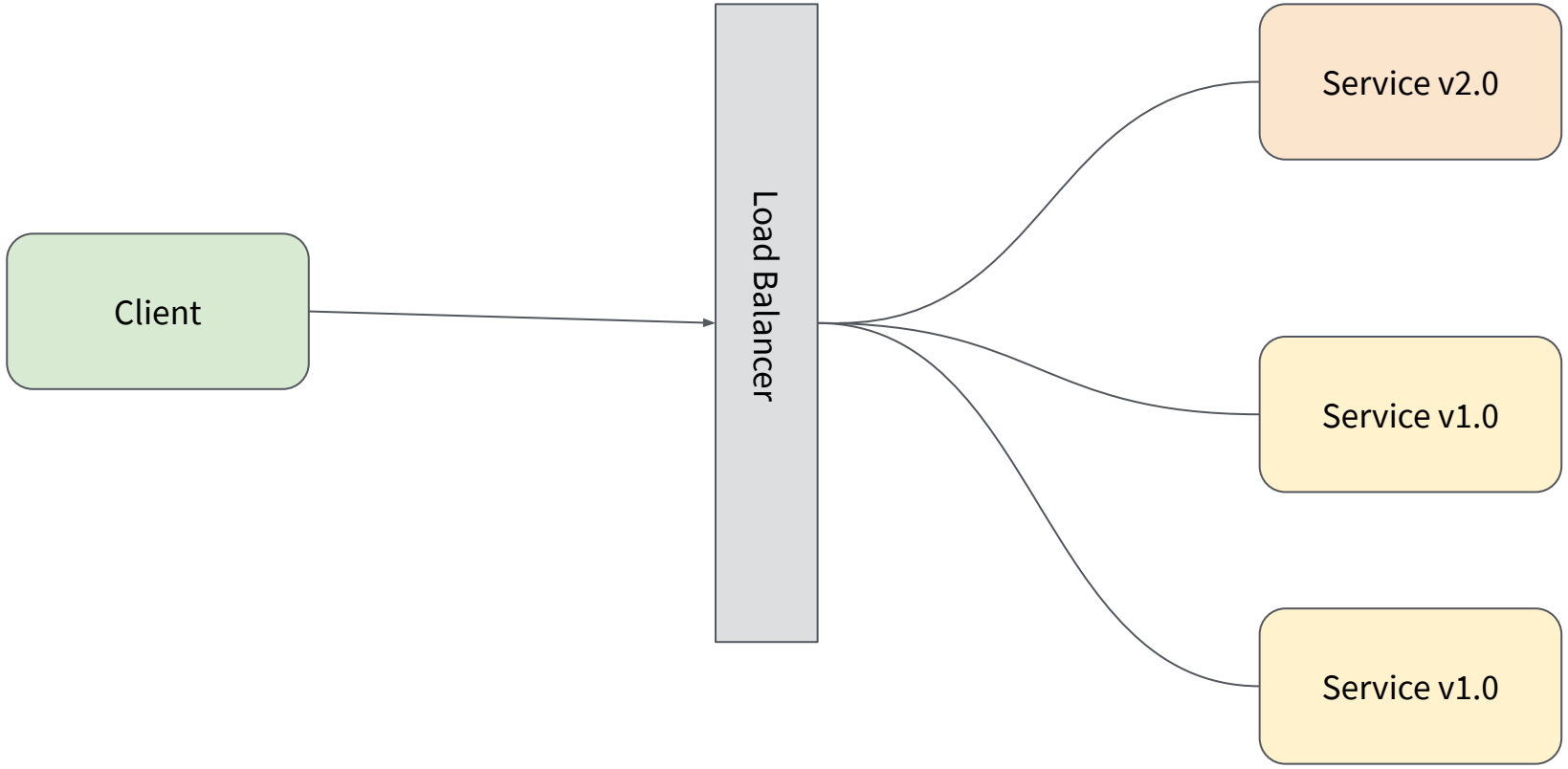
- No state persisted

Stateless Applications



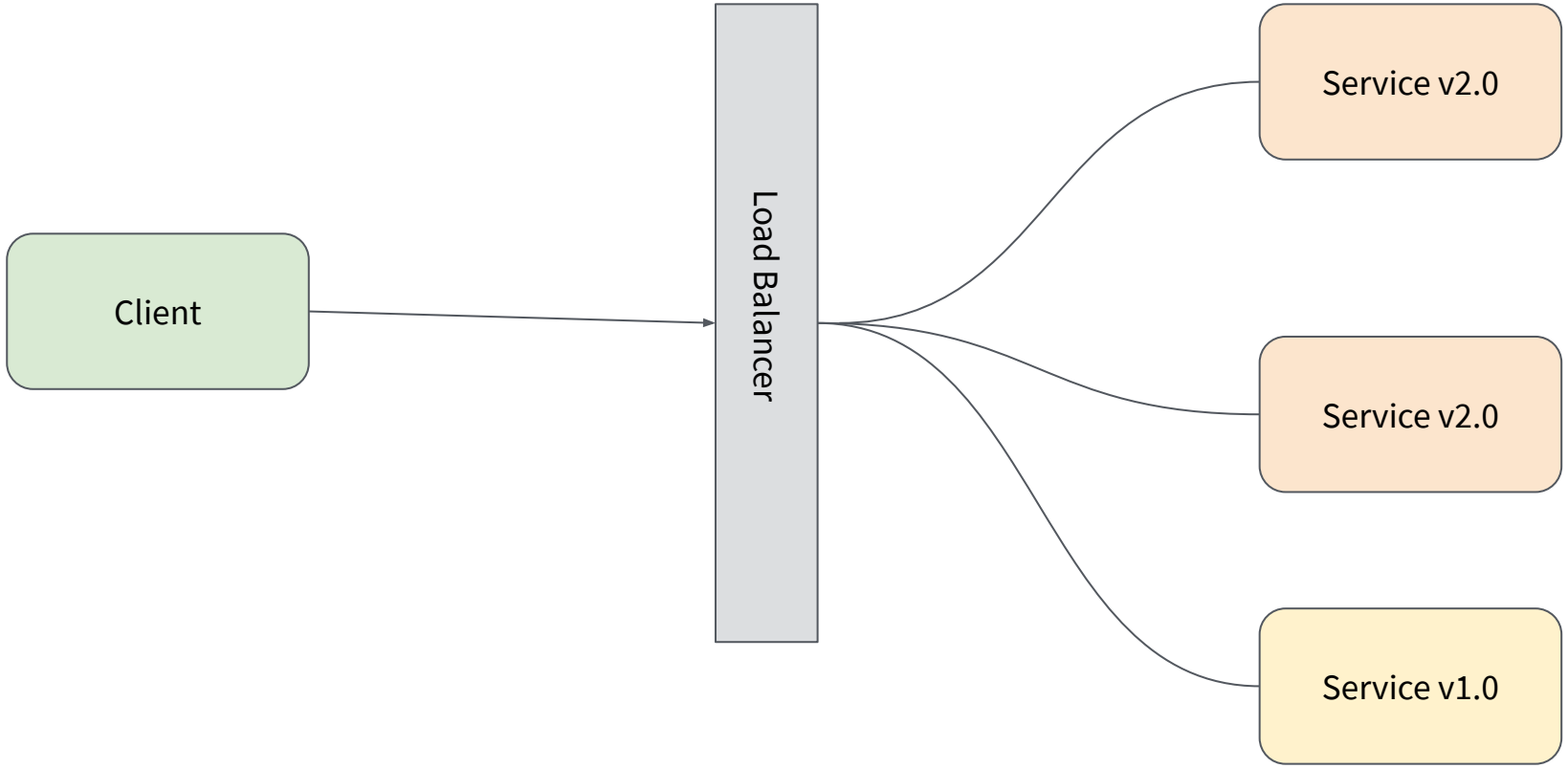
- No state persisted
- Easy to scale up / down

Stateless Applications



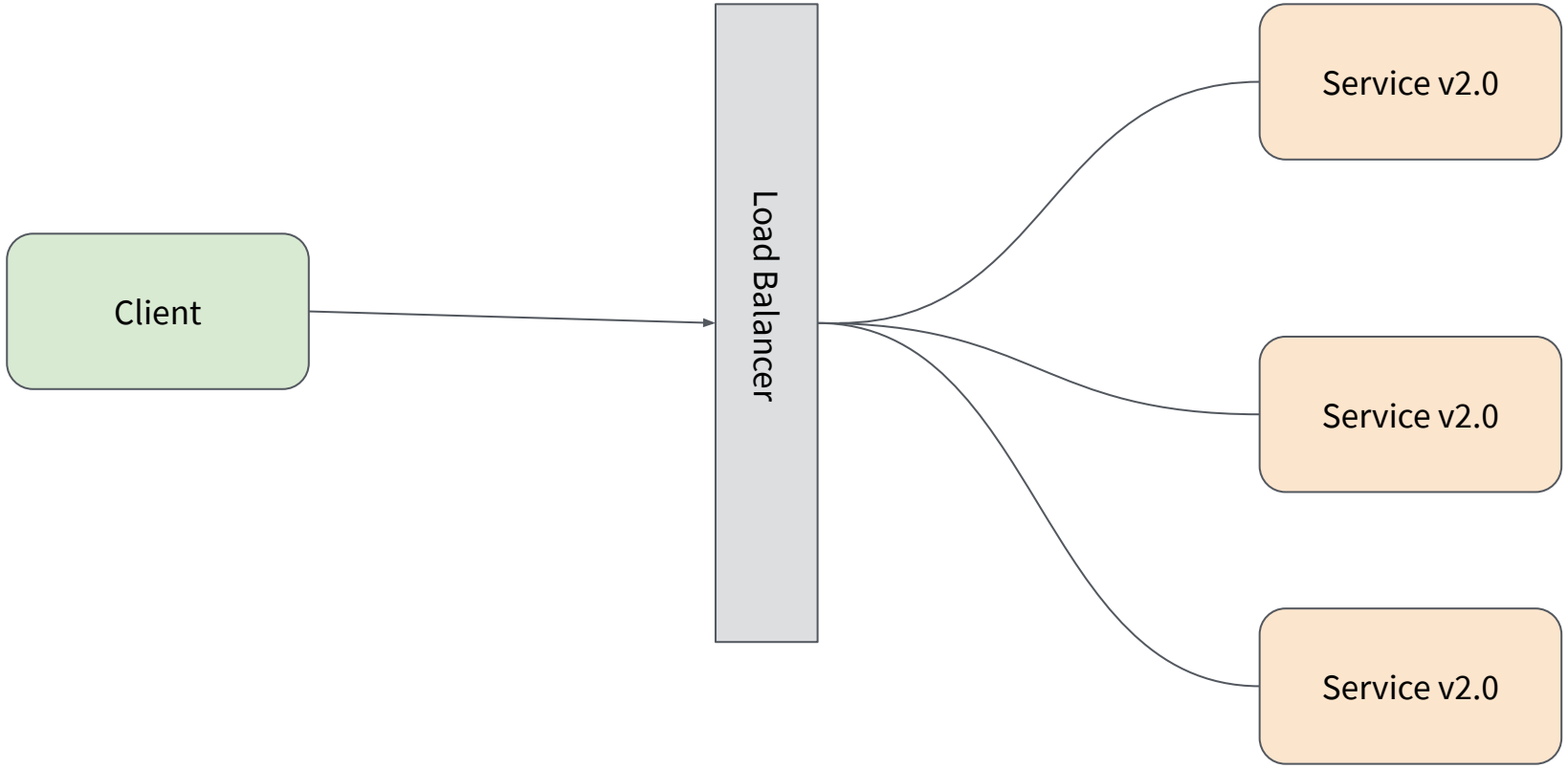
- No state persisted
- Easy to scale up / down

Stateless Applications



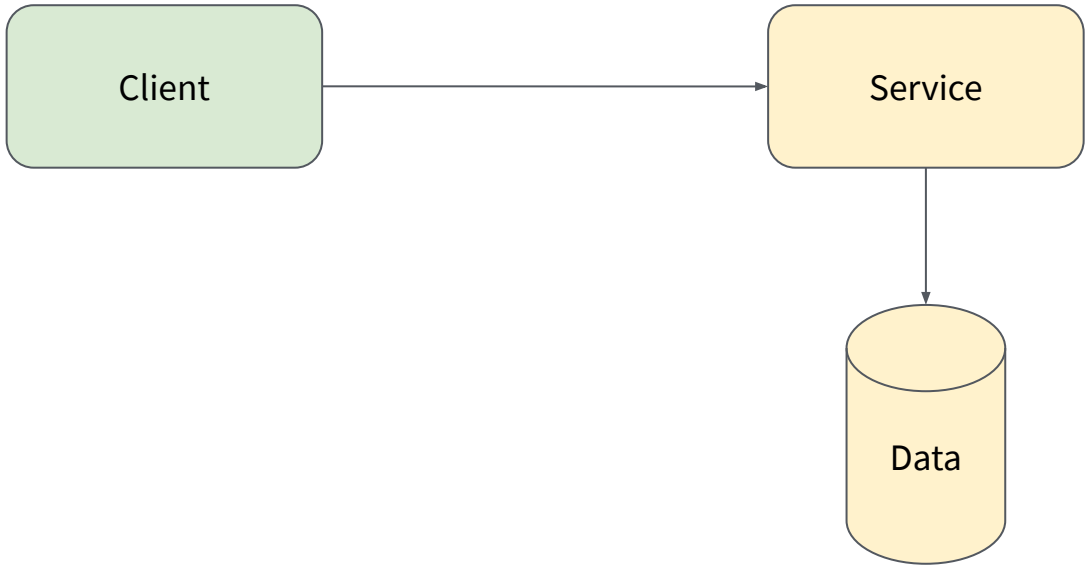
- No state persisted
- Easy to scale up / down

Stateless Applications

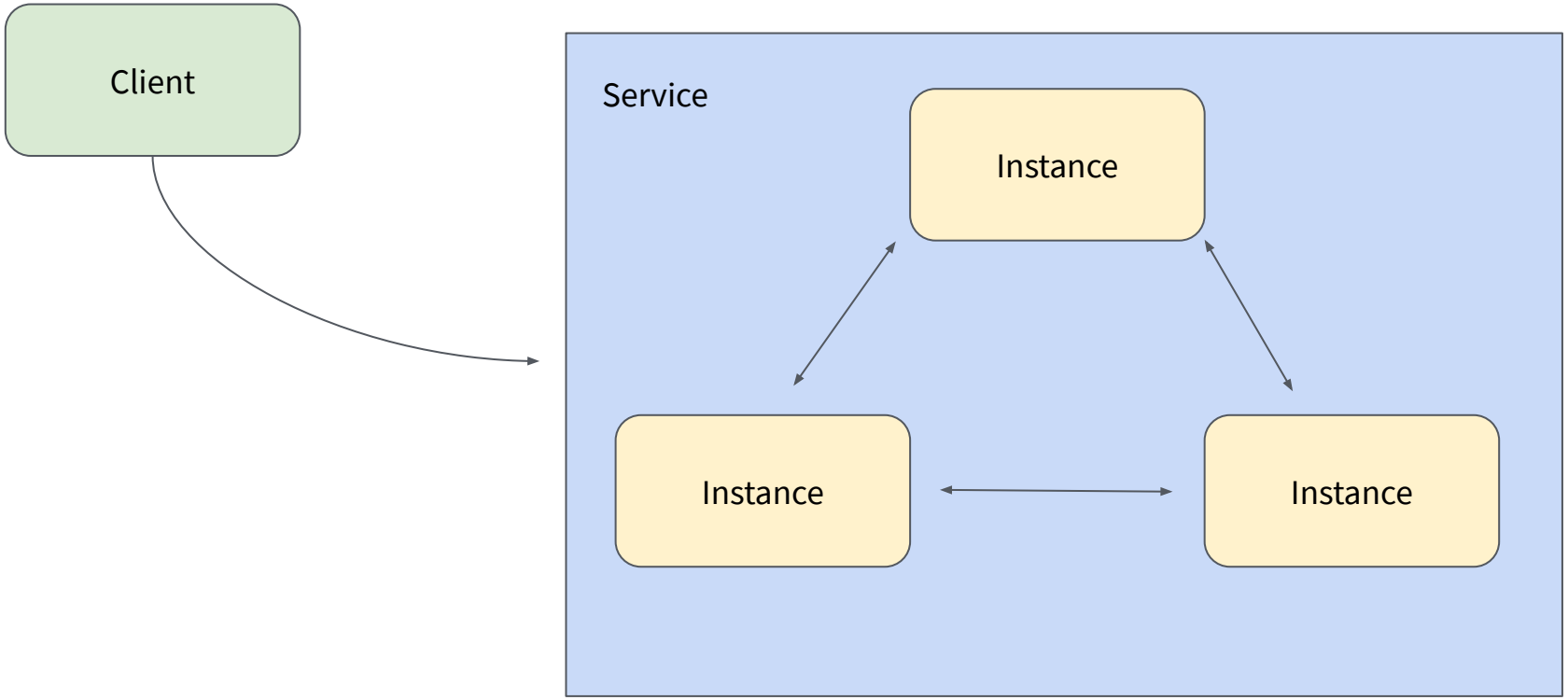


- No state persisted
- Easy to scale up/down
- Straightforward to upgrade

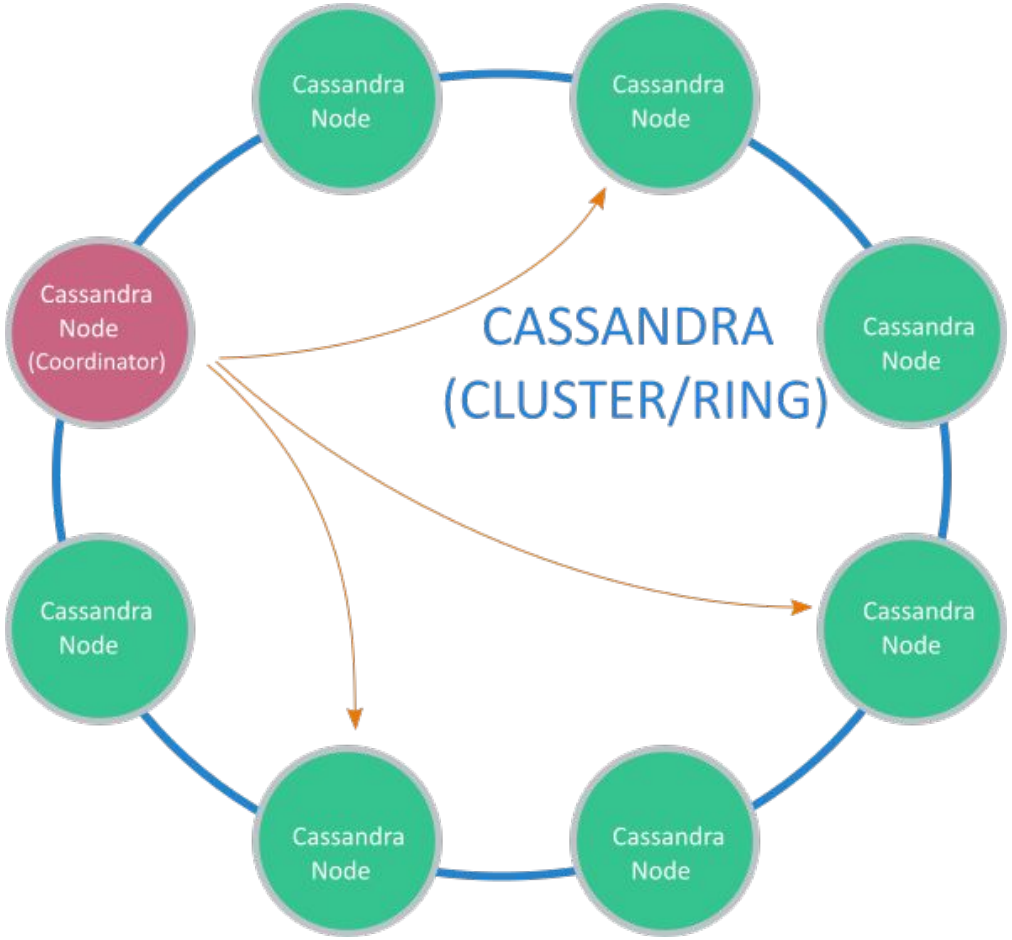
Stateful Applications



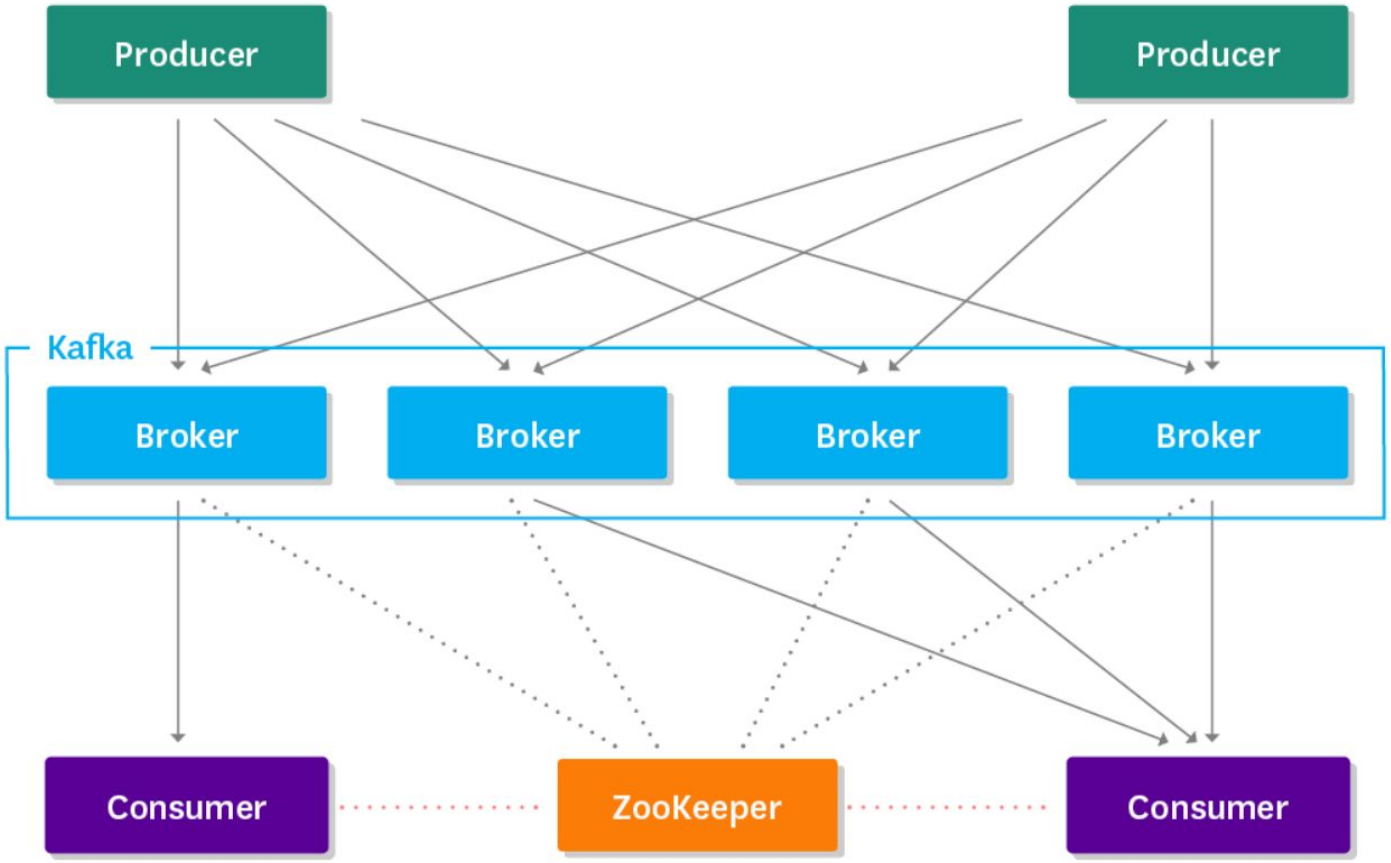
Stateful Applications



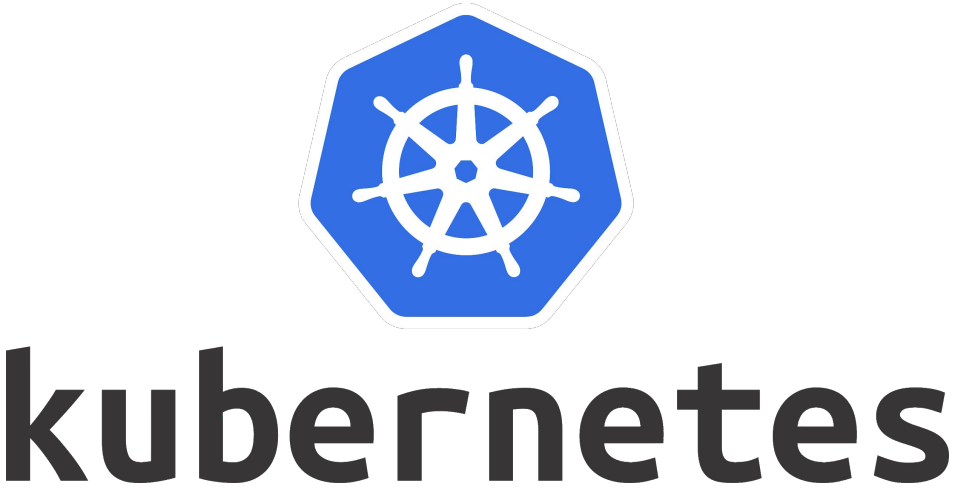
Distributed Stateful Applications



Distributed Stateful Applications



Kubernetes



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

Kubernetes - StatefulSets

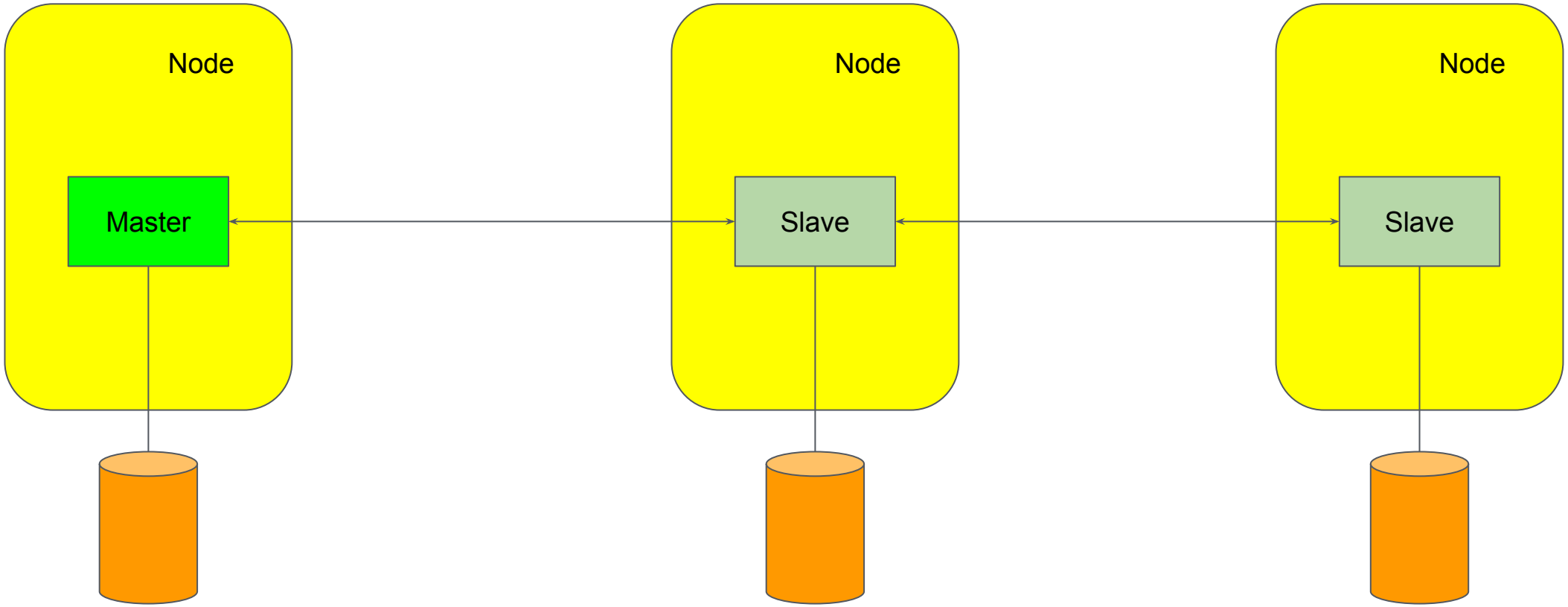
StatefulSets are valuable for applications that require one or more of the following:

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

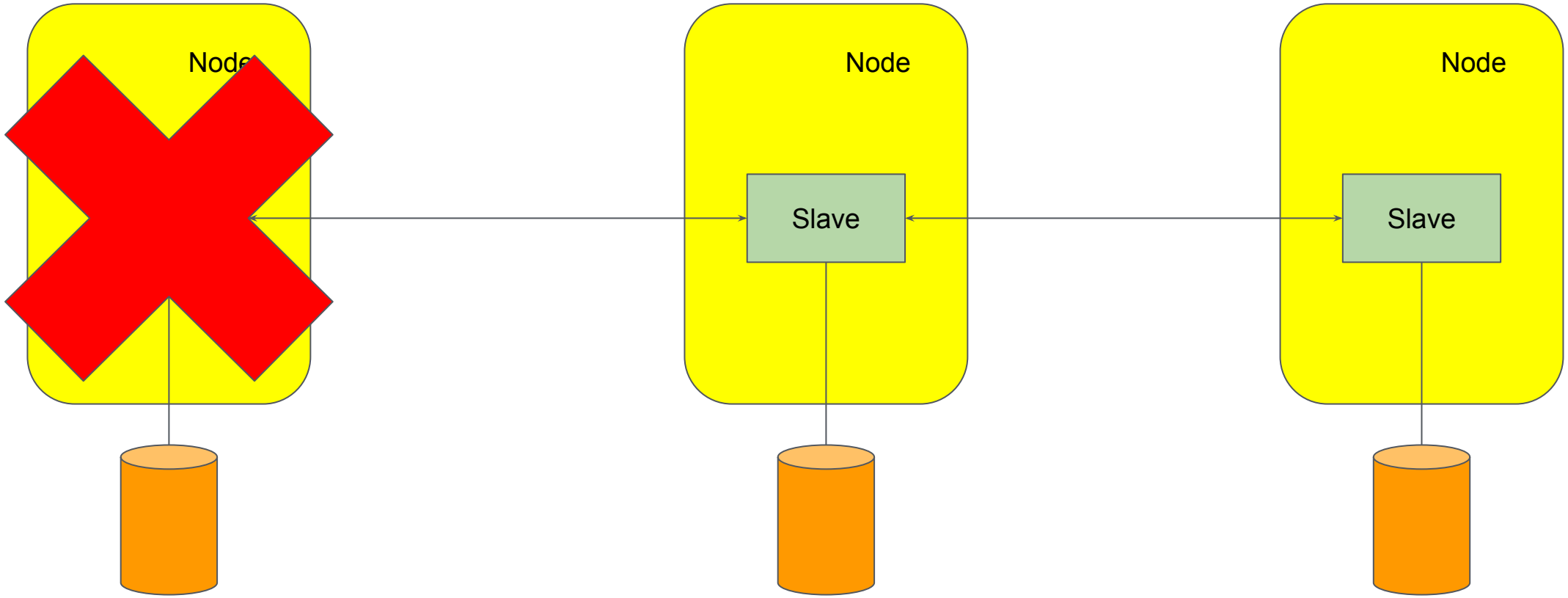
Kubernetes

kubernetes

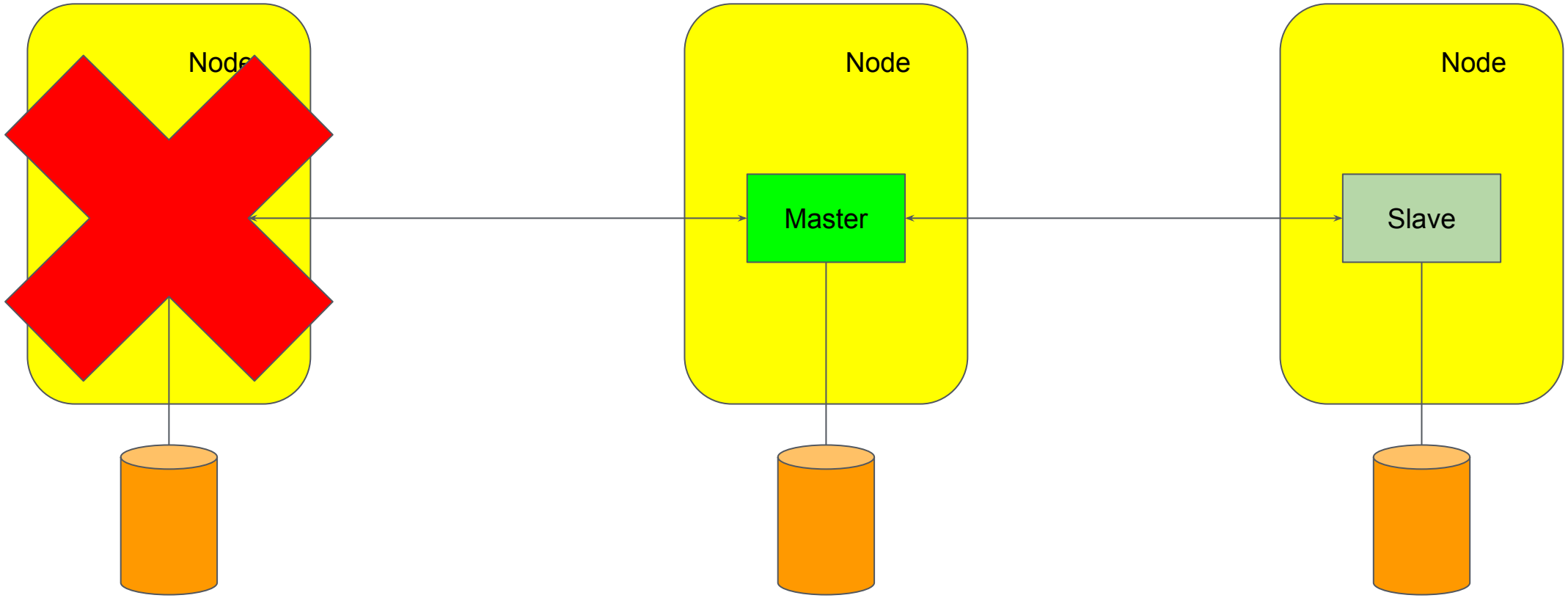
Kubernetes - StatefulSets



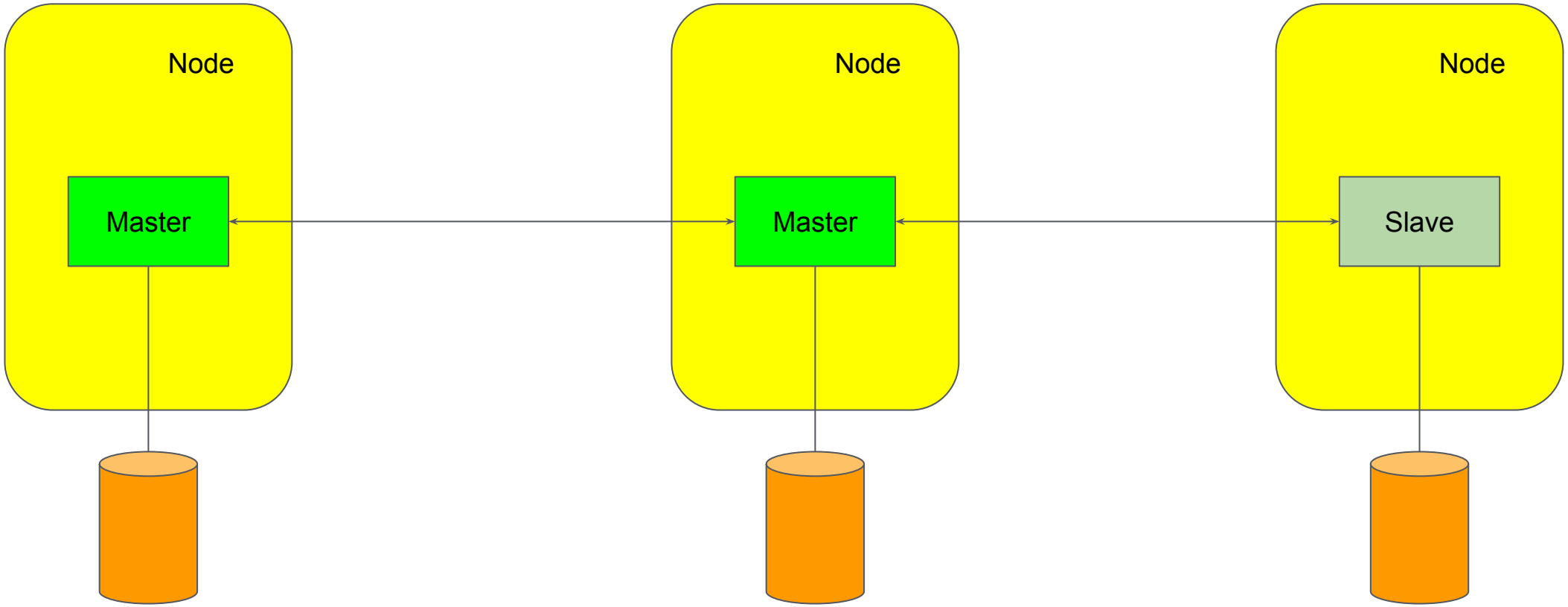
Kubernetes - StatefulSets



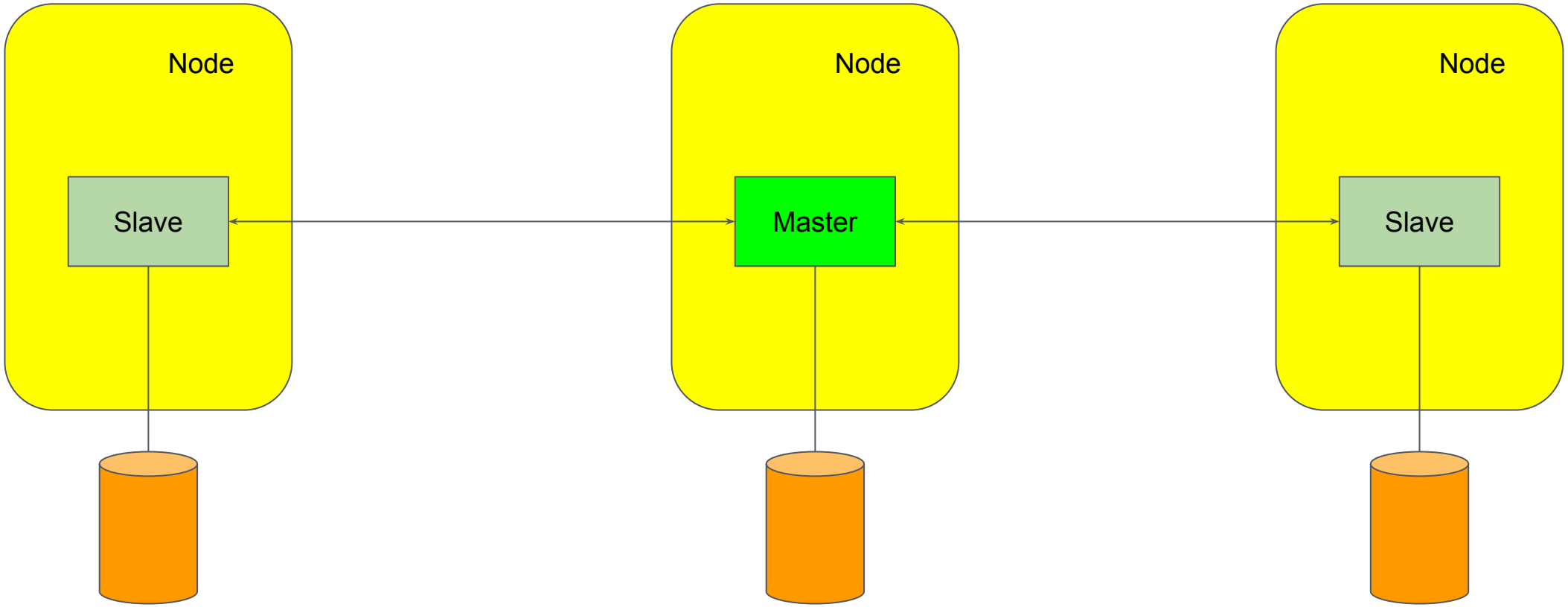
Kubernetes - StatefulSets



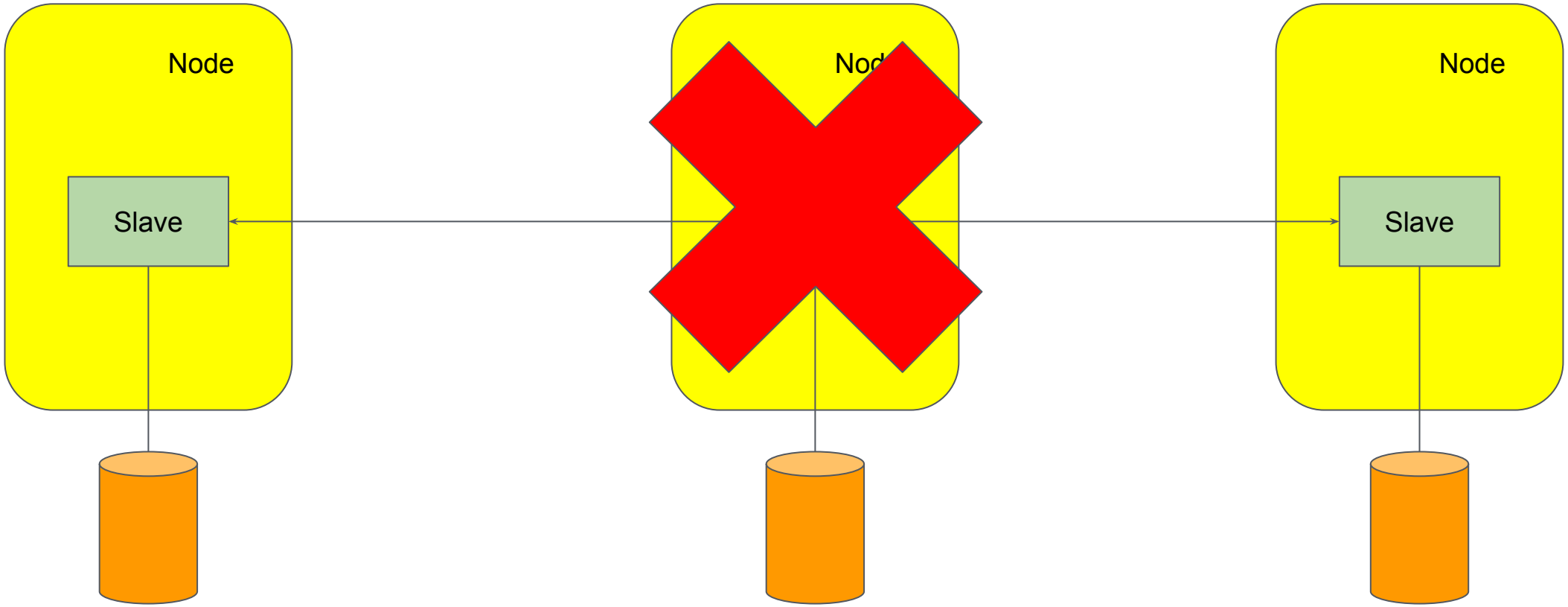
Kubernetes - StatefulSets



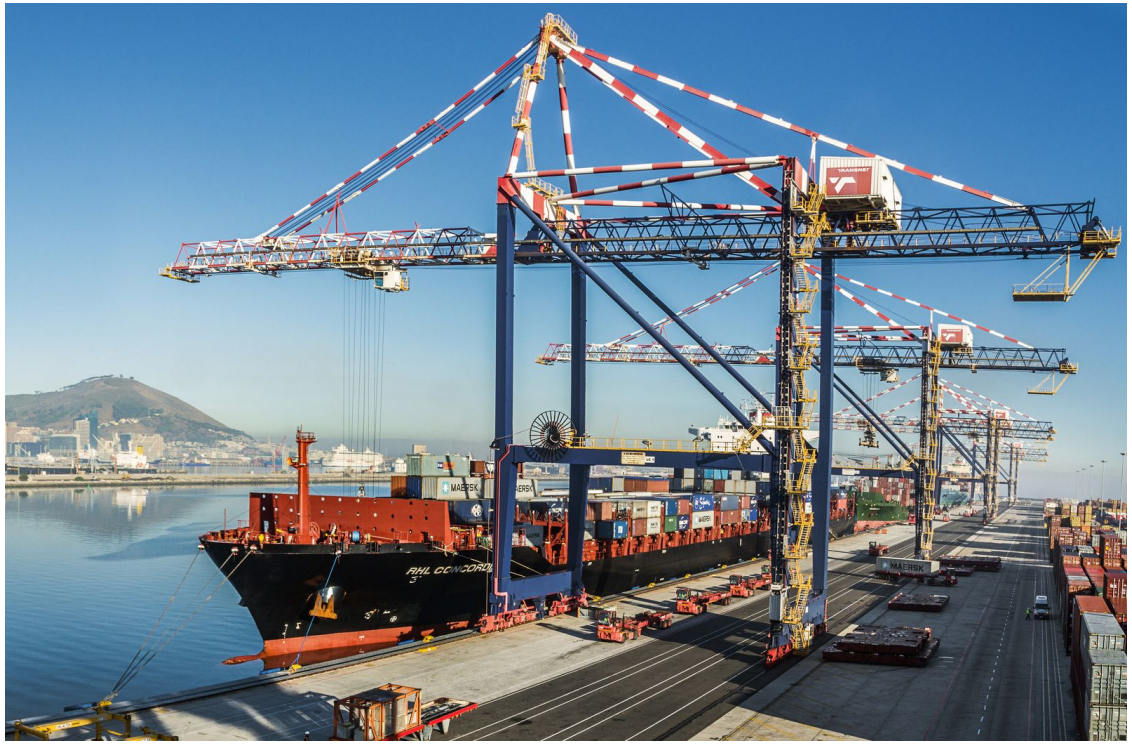
Kubernetes - StatefulSets



Kubernetes - StatefulSets

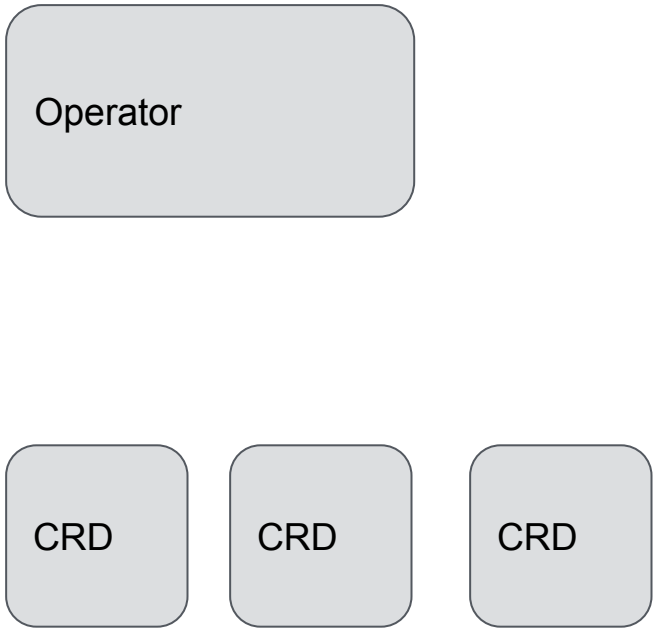


Operators



- 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

Operators



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

```
apiVersion:
mysql.presslabs.org/v1alph
a1
kind: MysqlCluster
metadata:
  name: my-cluster
spec:
  replicas: 2
  secretName: my-secret
```

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

Developing Operators

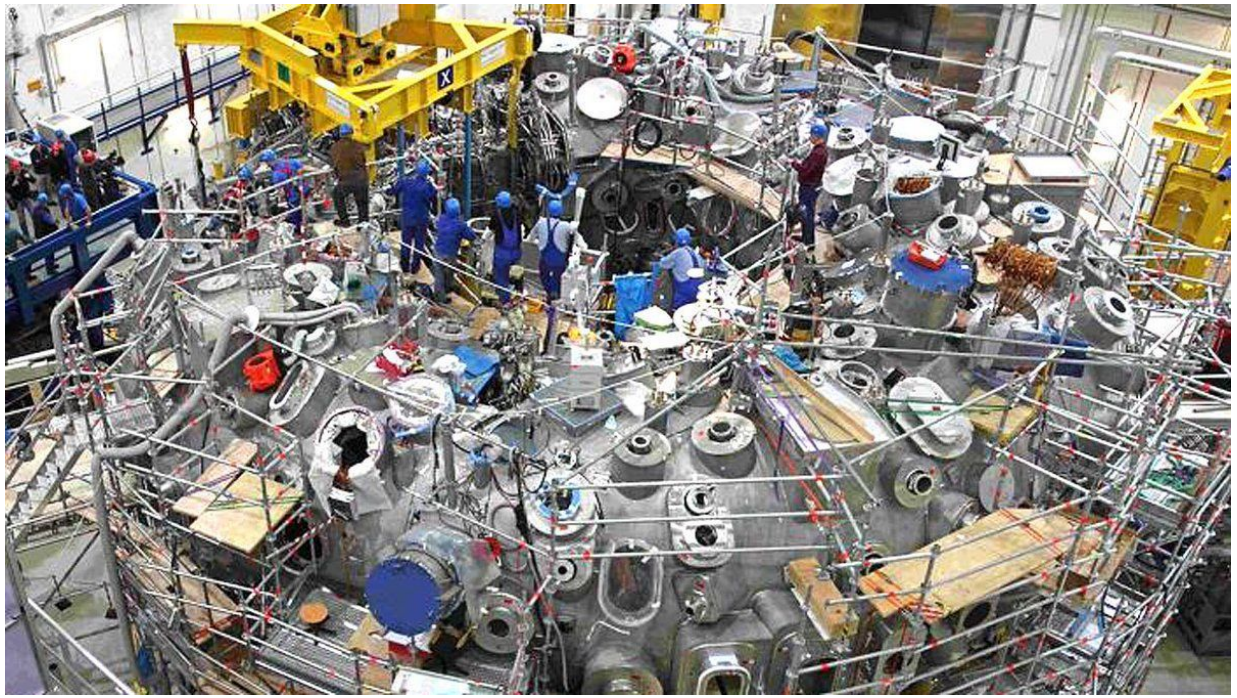
Operator Framework

- 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
- May require thousands of lines of code

KUDO

- Kubernetes Universal Declarative Operator
  - Universal operator configured via YAML
  - Encodes commonality and reuse between lifecycle operations
  - Provide ‘Plans’ as a way of sequencing and abstracting
- OS project licensed as Apache 2.0
- Open Development model
- “KUDO make me a sandwich”

Operator Development

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

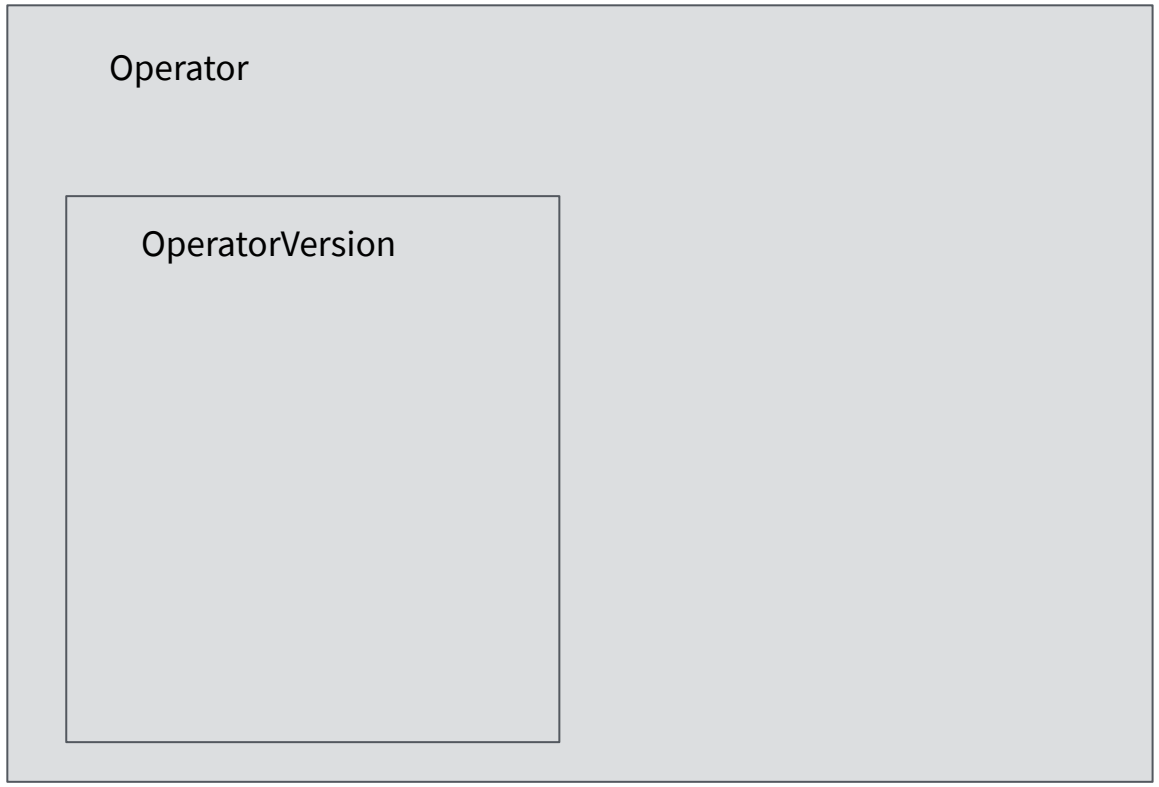
KUDO

- 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"

KUDO Concepts - ‘Operator’

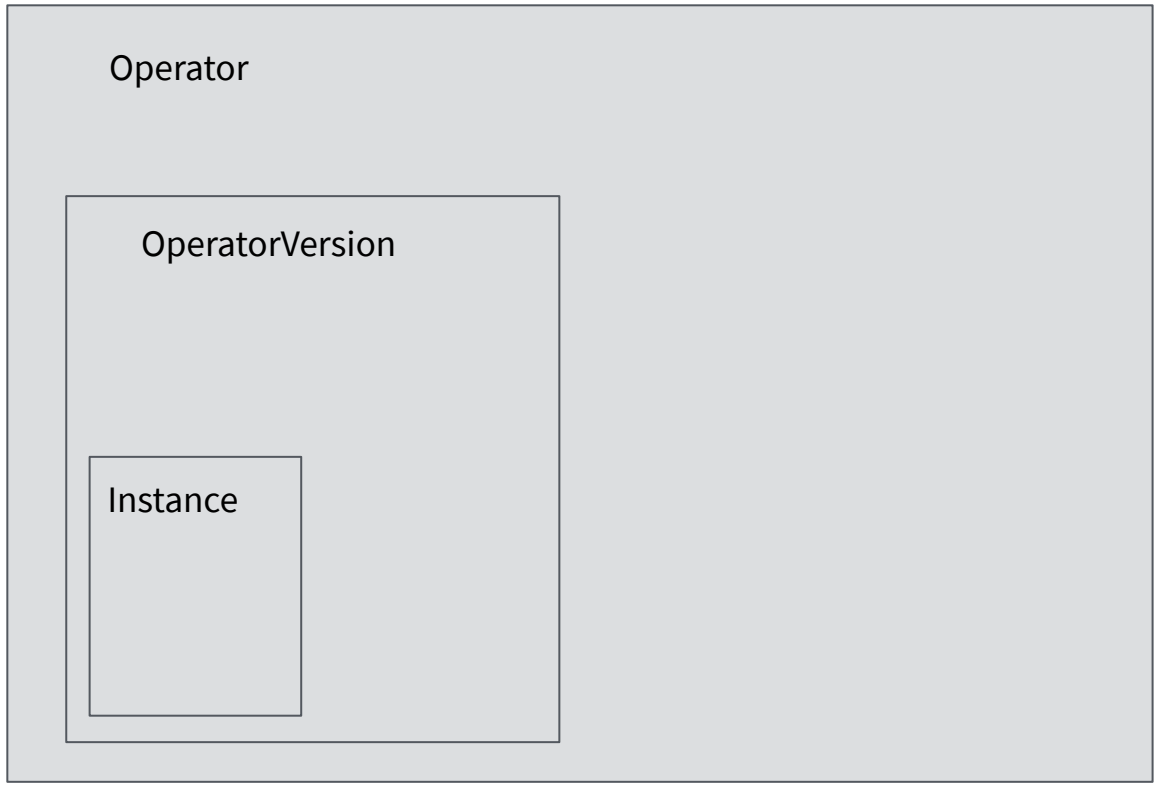
- High level description of a deployable service
- A deployable service can be anything that you’d want to run on your cluster
- Represented as a CRD object

KUDO Concepts - ‘OperatorVersion’



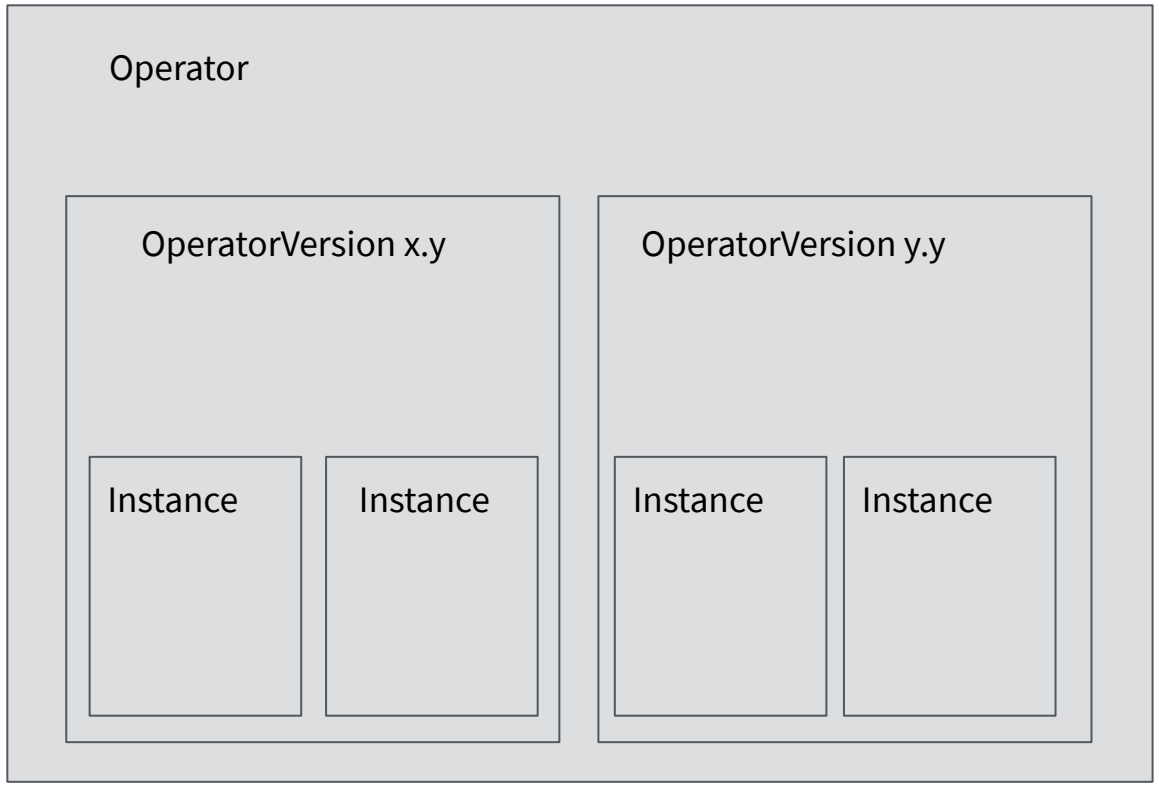
- Implementation of an Operator
- Specific version of a deployable application
- Contains parameters, objects, plans

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

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

Kudo Concepts - ‘Plan’

```
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

Kudo Concepts - ‘PlanExecution’

- Defines inputs and status of an instance’s executable plans
- Each execution of a Plan has its own CRD

```
Plan(s) for "mysql" in namespace "default":
.
└── mysql (Operator-Version: "mysql-0.1.0" Active-Plan: "mysql-restore")
    ├── Plan backup (serial strategy) [NOT ACTIVE]
    │   └── Phase backup (serial strategy) [NOT ACTIVE]
    │    └── Step backup (serial strategy) [NOT ACTIVE]
    │        ├── pv [NOT ACTIVE]
    │        ├── backup [NOT ACTIVE]
    │        └── cleanup [NOT ACTIVE]
    ├── Plan deploy (serial strategy) [NOT ACTIVE]
    │   └── Phase deploy (serial strategy) [NOT ACTIVE]
    │    └── Step deploy (serial strategy) [NOT ACTIVE]
    │        ├── deploy [NOT ACTIVE]
    │        ├── init [NOT ACTIVE]
    │        └── cleanup [NOT ACTIVE]
```

Kudo - CLI

- CLI extension to 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 Operator Examples - Plans

```yaml
plans:
  deploy:
    strategy: serial
    phases:
      - name: deploy
        strategy: serial
        steps:
          - name: deploy
            tasks:
            - deploy
          - name: init
            tasks:
            - init
          - name: cleanup
            tasks:
            - init
            delete: true
  backup:
    strategy: serial
    phases:
      - name: backup
        strategy: serial
        steps:
          - name: pv
            tasks:
            - pv
          - name: backup
            tasks:
            - backup
          - name: cleanup
            tasks:
            - backup
            delete: true
  restore:
    strategy: serial
    phases:
      - name: restore
        strategy: serial
```

Serial or parallel

Serial or parallel

Tasks can be reused in multiple plans

KUDO Operator Examples - Tasks

```yaml
tasks:
  deploy:
    resources:
    - service.yaml
    - pdb.yaml
    - configmap.yaml
    - statefulset.yaml
templates:
  service.yaml: |
    apiVersion: v1
    kind: Service
    metadata:
      name: svc
    spec:
      ports:
      - port: {{ .Params.BROKER_PORT }}
        name: server
      clusterIP: None
      selector:
        app: kafka
        instance: {{ .Name }}
  pdb.yaml: |
    apiVersion: policy/v1beta1
    kind: PodDisruptionBudget
    metadata:
      name: pdb
    spec:
      selector:
        matchLabels:
          app: kafka
          instance: {{ .Name }}
      minAvailable: 2
```

Go / Sprig templated K8s objects

KUDO Operator - Parameters

```yaml
parameters:
- name: BROKER_COUNT
  description: "Number of brokers spun up for Kafka"
  default: "3"
  displayName: "Broker Count"
- name: BROKER_CPUS
  description: "CPUs allocated to the Kafka Broker pods"
  default: "200m"
- name: BROKER_MEM
  description: "Memory (limit) allocated to the Kafka Broker pods"
  default: "200m"
- name: BROKER_PORT
  description: "Port brokers run on"
  default: "9093"
- name: KAFKA_NUM_PARTITIONS
  description: "Number of partitions for Kafka topics"
  default: "3"
- name: KAFKA_ZOOKEEPER_URI
  description: |
    host and port information for Zookeeper connection.
    e.g. zk:2181,zk2:2181,zk3:2181
  required: true
- name: KAFKA_ZOOKEEPER_PATH
  description: "Path inside of KAFKA_ZOOKEEPER_URI to host data"
  default: "/kafka"
- name: KAFKA_AUTO_CREATE_TOPICS_ENABLE
  default: "true"
```

Define parameter

Default value

KUDO Operator Examples - Instance

```yaml
apiVersion: kudo.k8s.io/v1alpha1
kind: OperatorVersion
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: kafka-2.11-2.4.0
  namespace: default
spec:
  serviceSpec:
  version: "2.11-2.4.0"
  connectionString: ""
  operator:
    name: kafka
    kind: Operator
  parameters:
  - name: BROKER_COUNT
    description: "Number of brokers spun up for Kafka"
    default: "3"
    displayName: "Broker Count"
  - name: BROKER_CPUS
    description: "CPUs allocated to the Kafka Broker pods"
    default: "200m"
  - name: BROKER_MEM
    description: "Memory (limit) allocated to the Kafka Broker
    default: "200m"
  - name: BROKER_PORT
    description: "Port brokers run on"
    default: "9093"
  - name: KAFKA_NUM_PARTITIONS
    description: "Number of partitions for Kafka topics"
    default: "3"
  - name: KAFKA_ZOOKEEPER_URI
    description: |
      host and port information for Zookeeper connection.
      e.g. zk:2181,zk2:2181,zk3:2181
    required: true
```

Currently a single CRD for Instantiations of a OperatorVersion

Reference OperatorVersion to use

Define parameters for instance

DEMO

KUDO Futures - Dynamic CRD's

- Create new CRD's during runtime operation
- Update existing CRD's

```
$ kubectl kudo update-parameter-trigger --framework kafka \
  --framework-version 1.1 --parameter IMAGE_TAG blue-green

Update successful.  IMAGE_TAG updates will now trigger the
`blue-green` plan

$ kubectl apply -f new-plan.yaml
plan.kafka.kudo.dev/latest blue-green created
```

KUDO Futures - Dynamic CRD's

- Represent components as first-class Kubernetes objects
- Represent *operations* as first-class Kubernetes objects
  - Kind: Kafka
  - Kind: Topic
  - Kind: User
  - Backup
    - maxAge: 24h
  - Backup / Restore
  - Index

KUDO Futures - Framework Extensions

Framework Developer Maintained

ACME Corp Maintained

MySQL

"Standard" infrastructure, plans, CRDs, etc.

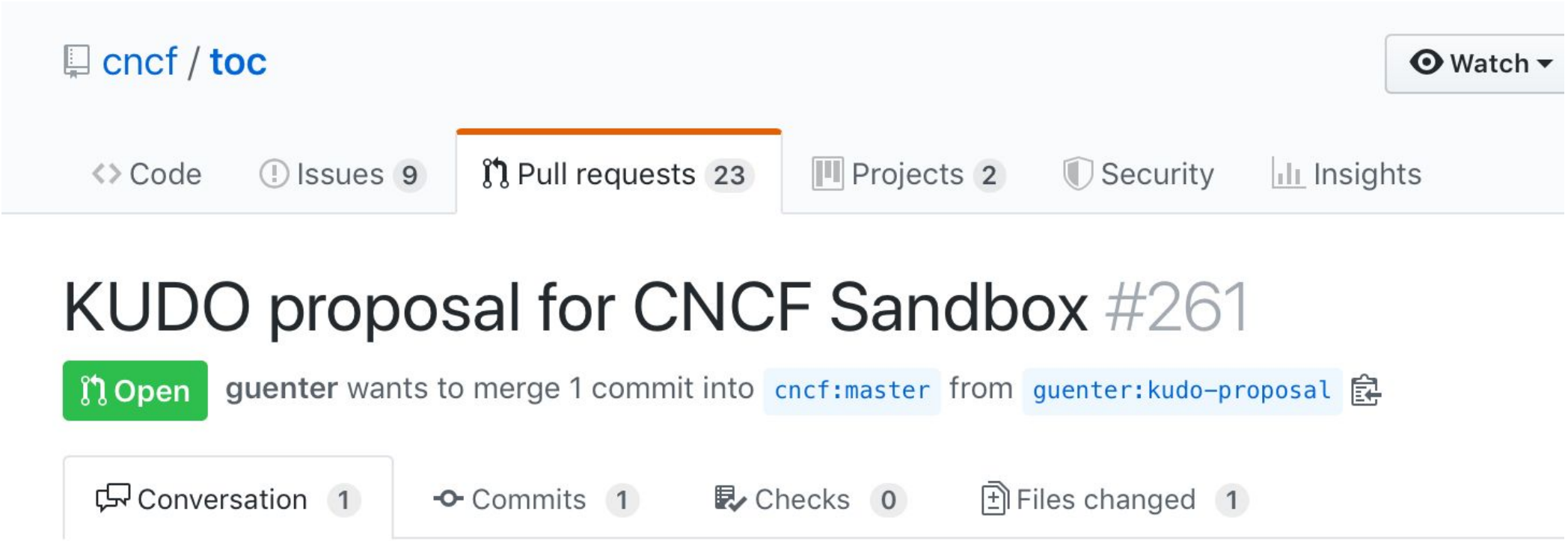
MySQL + GKE

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

ACME Corp

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

Future - CNCF


cncf / toc
Watch
Code
Issues 9
Pull requests 23
Projects 2
Security
Insights
KUDO proposal for CNCF Sandbox #261
Open
guenter wants to merge 1 commit into cncf:master from guenter:kudo-proposal
Conversation 1
Commits 1
Checks 0
Files changed 1

Get Involved !

https://kudo.dev/

https://github.com/kudobuilder/kudo

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

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

Community Meeting - weekly Thursdays 10am PT