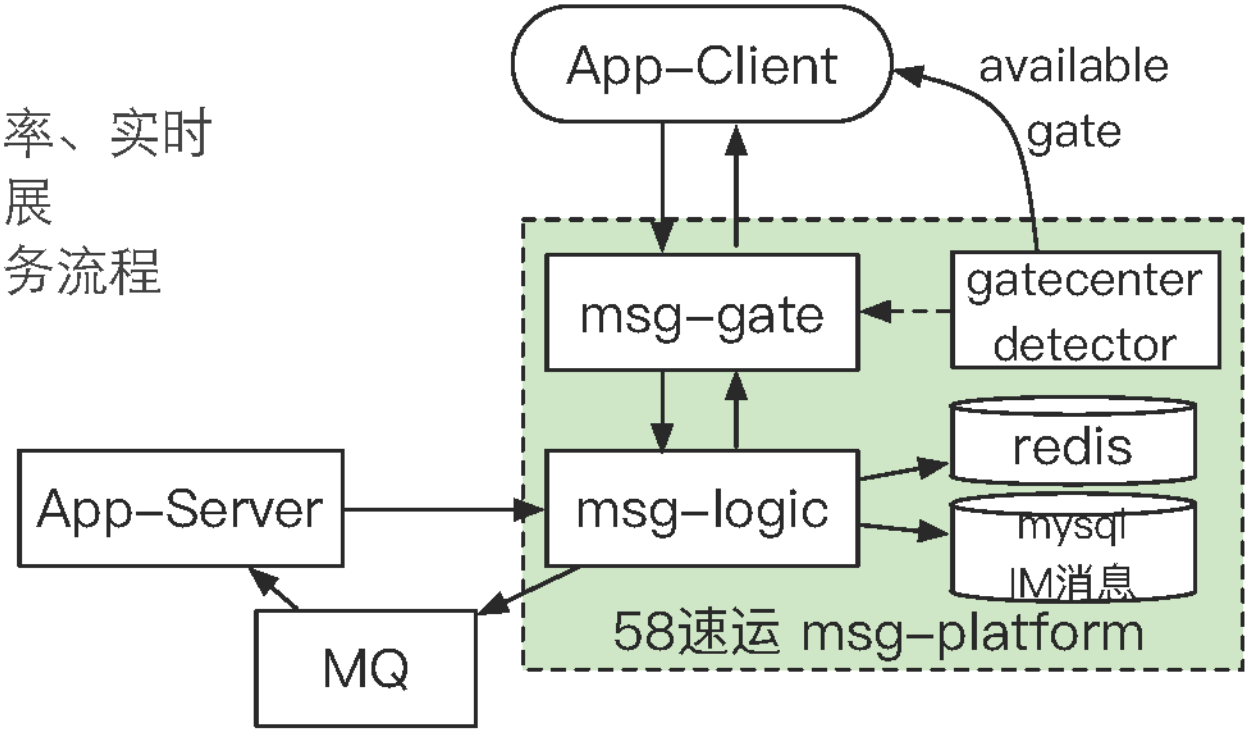
总结

- 分层设计
- 高性能
- 高可用
- 高到达率、实时
- 高可扩展
- 核心业务流程

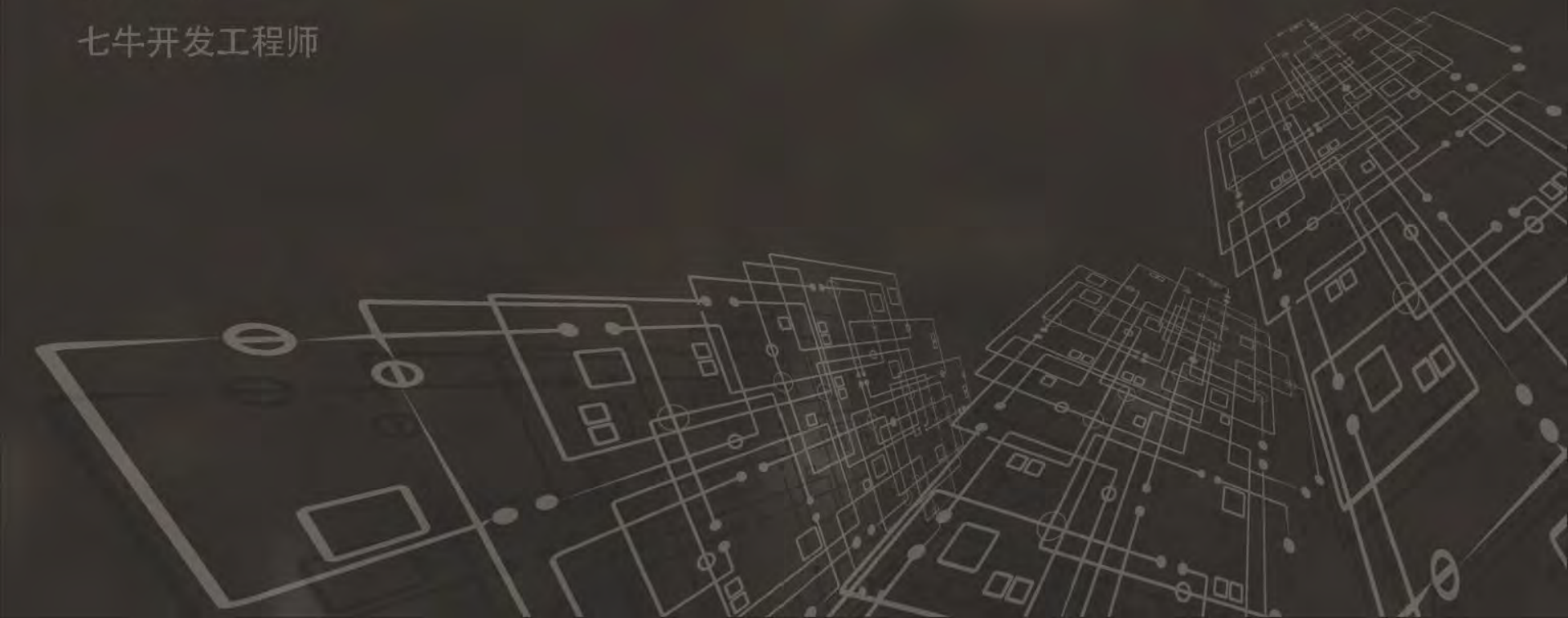


2017 Software Architecture Summit

基于 Kubernetes 的高可用 MySQL 微服务实践

张翼飞

七牛开发工程师



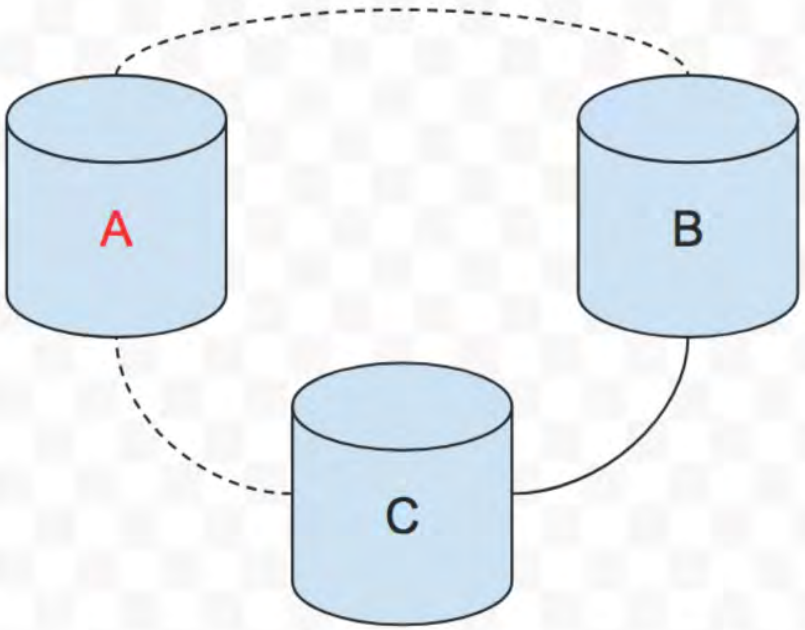
主题

- 挑战
- MySQL 集群常见故障
- MySQL 集群数据备份/恢复
- Operator 简介
- 关键问题
- 服务保障
- 架构设计
- 交付
- 总结

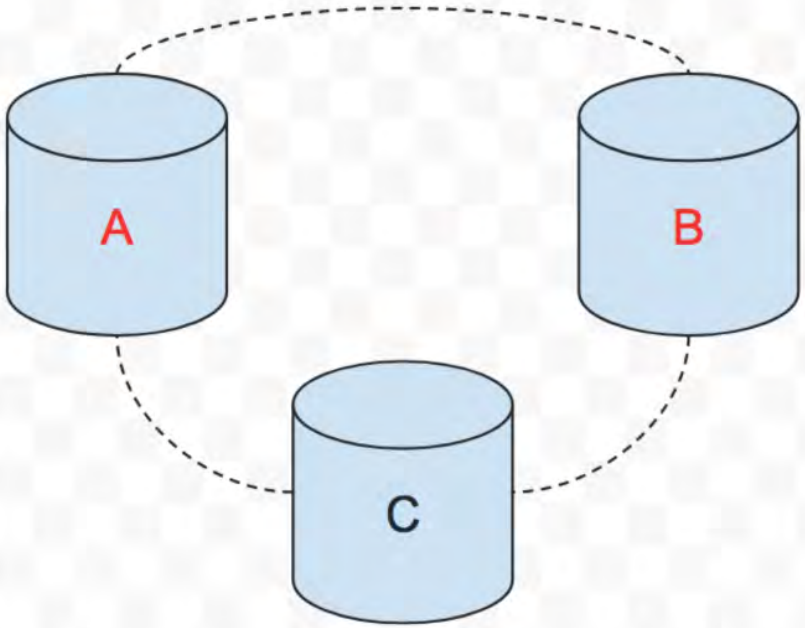
挑战

- MySQL 服务高可靠
- MySQL 服务高可用
- 基于 Kubernetes
- MySQL 服务自运维
- 支持多租户
- 易于部署和升级

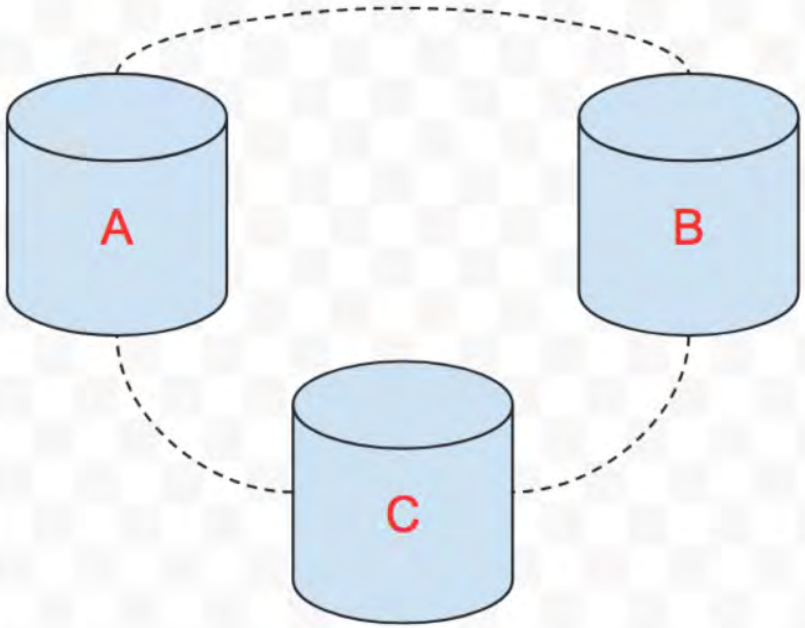
MySQL 集群常见故障_一个节点停止服务



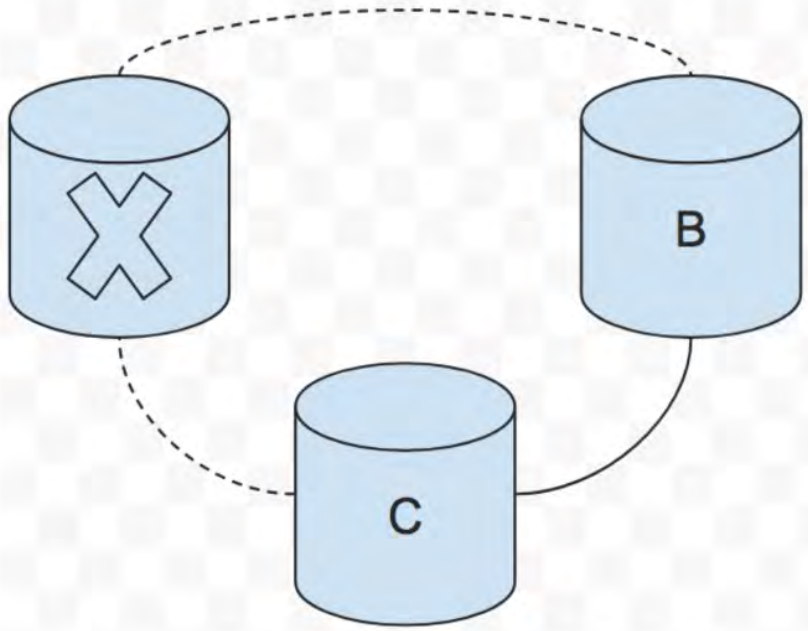
MySQL 集群常见故障_两个节点停止服务



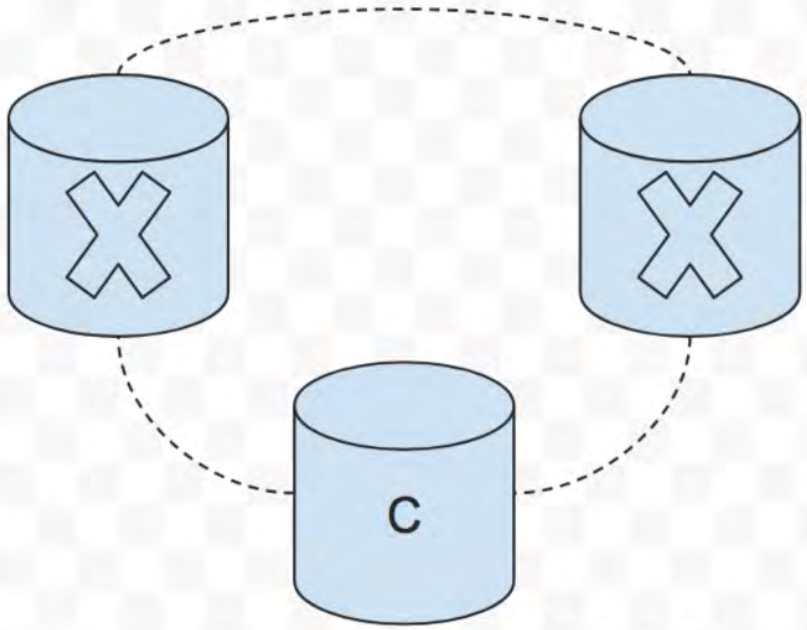
MySQL 集群常见故障_全部节点停止服务



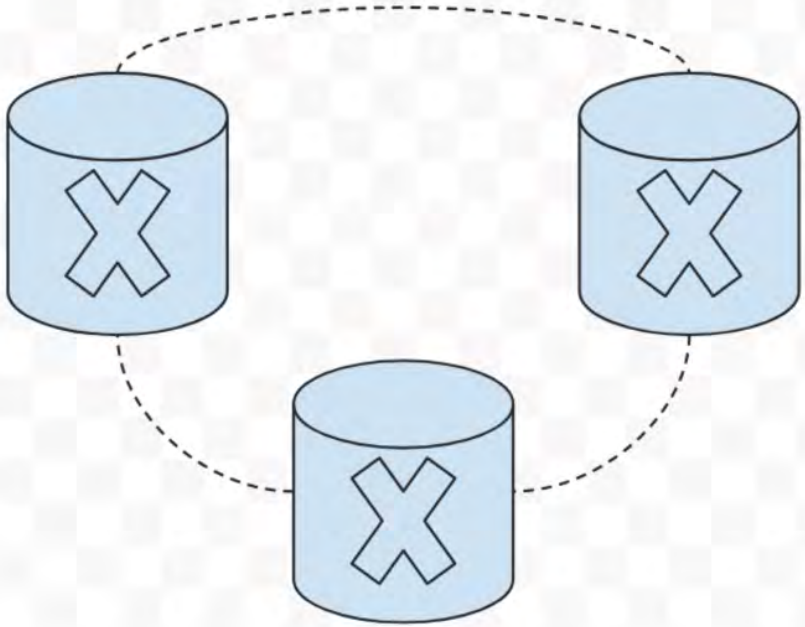
MySQL 集群常见故障_一个节点挂掉



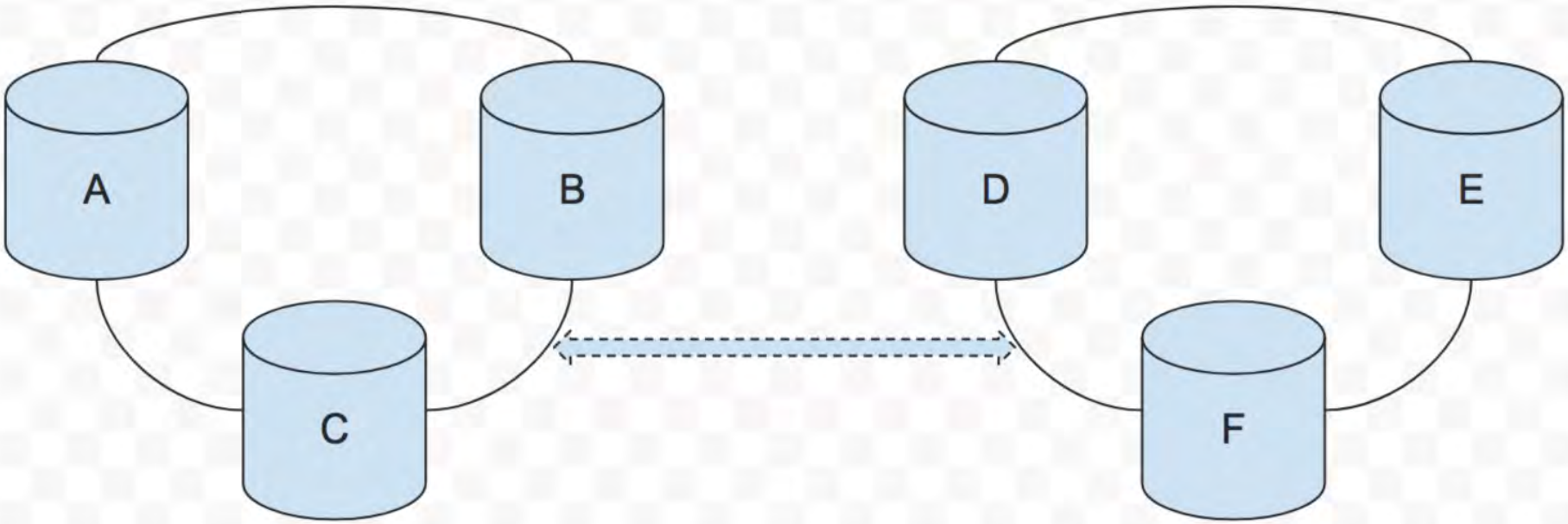
MySQL 集群常见故障_两个节点挂掉



MySQL 集群常见故障_全部节点挂掉



MySQL 集群常见故障_脑裂



MySQL 集群数据备份/恢复_如何备份

- 通过 XtraBackup 工具实现「热备份」

MySQL 集群数据备份/恢复_如何恢复

- 备份当前状态数据
- 暂停全部节点服务
- 选取某个节点作为 bootstrap 节点
- 在该 bootstrap 节点上恢复数据
- 启动其他的节点

Operator 简介

Operator 是什么?

针对特定应用的 Controller，封装应用的领域内知识。

Operator 简介

Operator 解决的问题是什么?

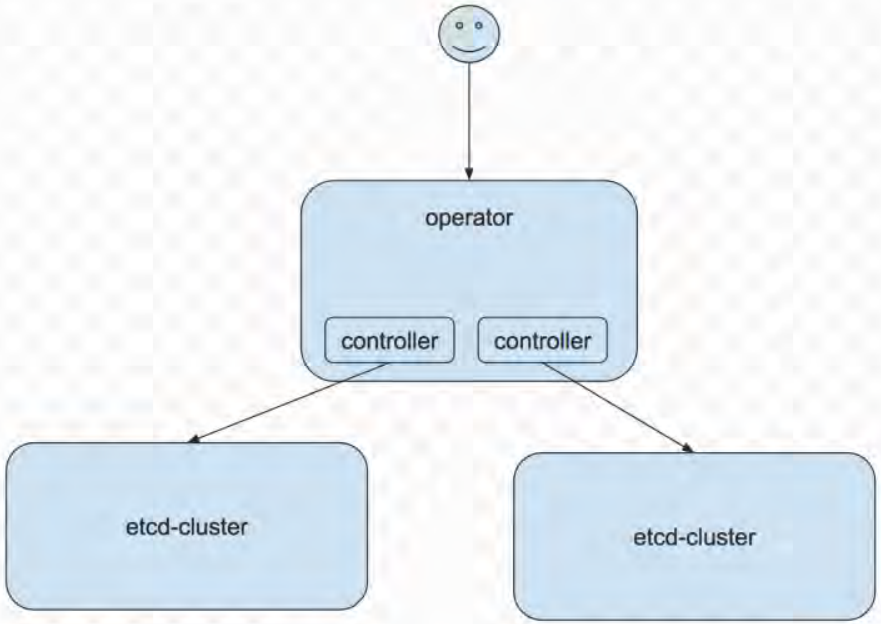
如何为用户提供易用的基于 Kubernetes 的复杂应用。

Operator 简介

Operator 的设计原则有哪些?

- Operator 仅通过一个简单的 Deployment 部署，且无需其他操作
- Operator 需要在 Kubernetes 中创建一个 Resource 表征应用
- Operator 需要使用 Kubernetes 中现有的服务来管理应用
- Operator 需要设计为它的停止和删除不影响已部署的应用
- Operator 需要支持应用的升级

Operator 简介_etcd-operator 架构

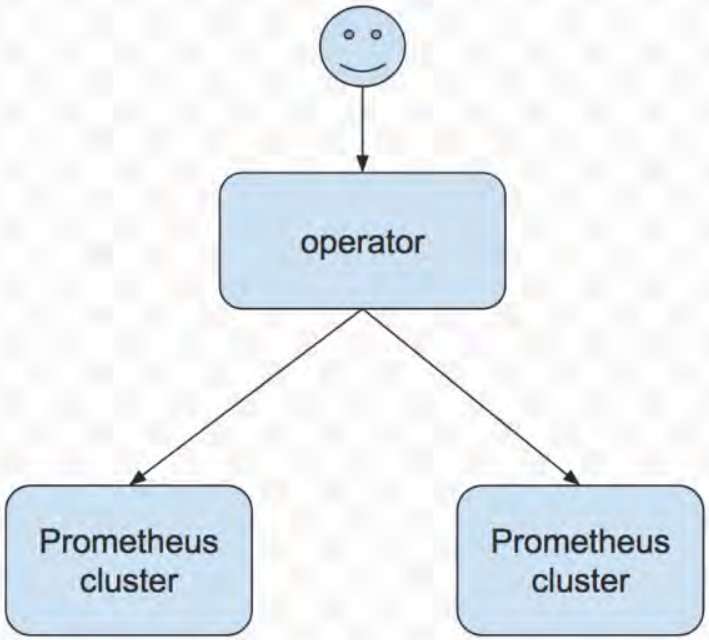


Operator 负责：

- 创建/删除 etcd 集群
- 扩缩容 etcd 集群
- 恢复 etcd 集群
- 备份/恢复 etcd 集群
- 升级 etcd 集群

- Operator 为每个 cluster 创建一个 Controller 管理该 cluster 的生命周期
- Controller 在该 cluster 生命周期内以 goroutine 方式持续运行
- Controller 定期 poll Pod 状态来做故障检测和自动处理

Operator 简介_prometheus-operator 架构



Operator 负责：

- 创建/删除 Prometheus 服务
- 创建/删除 ServiceMonitor 服务
- 创建/删除 AlterManager 服务

- Operator 使用 StatefulSet 创建集群
- 定时 re-list objects
- 所有监听到的事件入 Queue 串行处理

关键问题_如何控制复杂度

- 分层设计
- 状态驱动
- 机制与策略分离
- 服务解耦

关键问题_如何确保服务状态符合期望

- Operator 不断驱动服务「实际状态」转变为「期望状态」
- 服务的状态信息存储在 Kubernetes 的 objects 中
- Operator 定时 re-list objects
- Operator 根据 object 的状态，触发相应的任务
- Controller 收集状态信息并反馈给 Operator
- Operator 修改 object 状态

关键问题_如何控制不同逻辑之间的并发执行问题

- 由 Controller 管理任务调度
- 同步执行任务

关键问题_如何升级服务

- 使用 Kubernetes 特性，将「升级服务」转变为「升级 SPEC」

关键问题_Kubernetes 的环境限制

- 「Pod 异常」是正常情况，且是常态
- 「网络异常」是正常情况，且非常态
- 「存储异常」是正常情况，且非常态

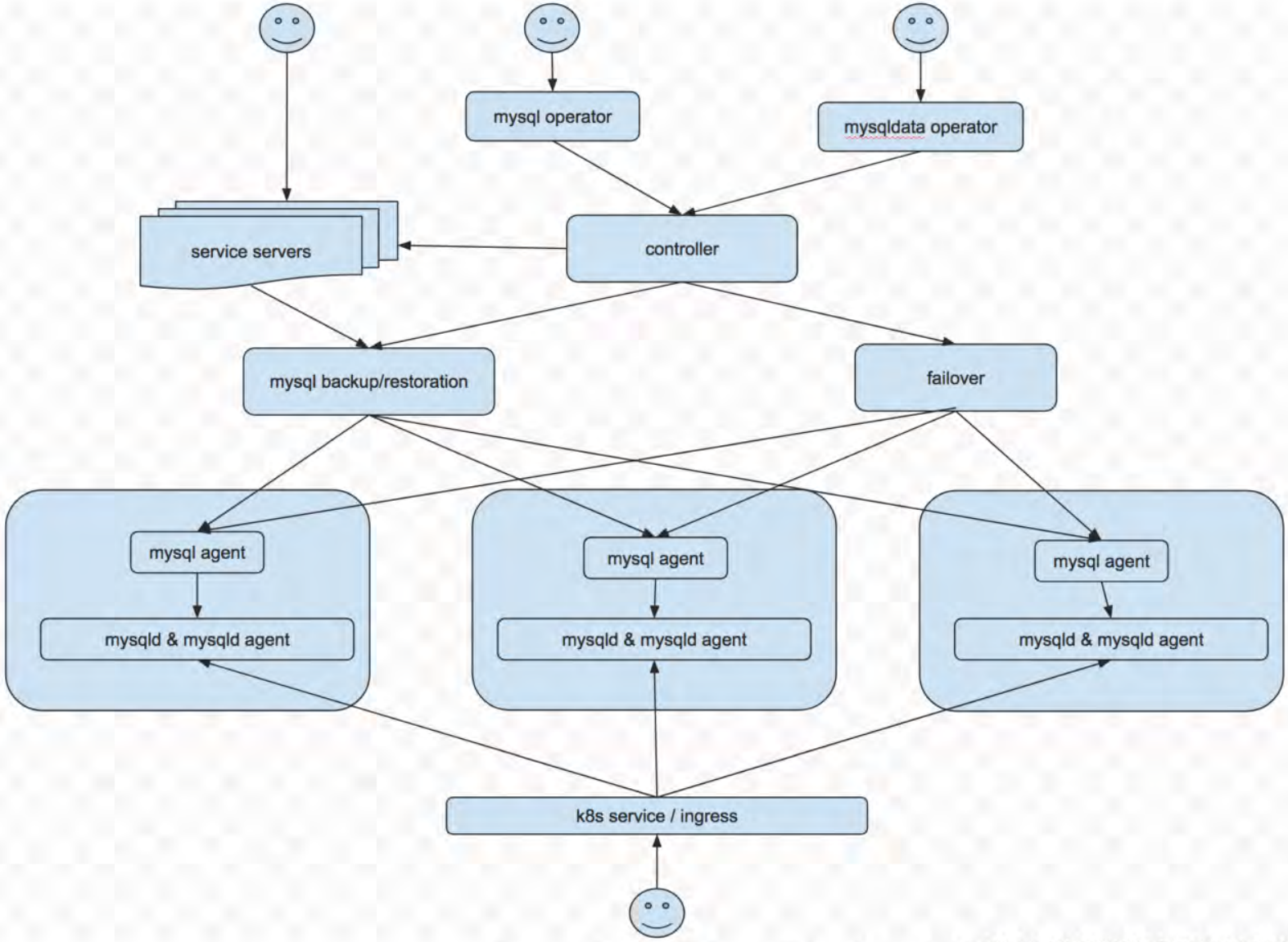
服务保障_高可靠

- 数据冗余
- 数据定时/手动备份
- 数据持久化在 PV 上
- 并发逻辑串行化调度

服务保障_高可用

- 服务冗余
- 状态驱动
- MySQL Pod 通过 StatefulSet 方式部署
- 每个 MySQL 集群自治

架构设计


mysql operator
mysqldata operator
service servers
controller
mysql backup/restoration
failover
mysql agent
mysqld & mysqld agent
mysql agent
mysqld & mysqld agent
mysql agent
mysqld & mysqld agent
k8s service / ingress

架构设计_OperatorInterface

```
type CRD struct {
        Name    string // format: Plural.Group
        Kind    string
        Plural  string
        Group   string
        Version string
        Scope   apiextensionsv1beta1.ResourceScope

        Obj             runtime.Object
        ObjList         runtime.Object
        SchemeBuilder func(*runtime.Scheme) error
}

type OperatorInterface interface {
        // CreateCRD creates the CRD resource.
        CreateCRD(*CRD) error
        // DeleteCRD deletes the CRD resource.
        DeleteCRD(name string, options *metav1.DeleteOptions) error
        // GetCRD gets the CRD resource.
        GetCRD(string) (*apiextensionsv1beta1.CustomResourceDefinition, error)

        // WatchEvents starts the controller to handle with the CRD's
        // ADD/DELETE/UPDATE events.
        WatchEvents(context.Context, *CRD) error
}
```

架构设计_ControllerInterface

```
type ControllerInterface interface {
        // Init initialize the controller.
        Init() error

        // Run starts the loop.
        Run() error

        // AddTask adds a task for the controller.
        AddTask(Task) error
        // PopTask pops a task for the controller to handle with.
        PopTask() (Task, error)

        // Feedback reports the current status of the MySQL Cluster to
        // the MySQL operator.
        Feedback(*Status) error

        // RestoreCluster restores the MySQL Cluster.
        RestoreCluster(*FaultInfo) error

        // RestoreData restores the data of the MySQL Cluster.
        RestoreData(*Dataset) error

        // ScheduleBackup implements the scheduled backup task.
        ScheduleBackup(*ScheduleInfo) error

        // BackupOnce does a backup operation.
        BackupOnce() error
}
```

架构设计_FailoverInterface

```
type FailoverInterface interface {
        // Init initialize the worker and create MySQL Cluster.
        Init(*FailoverConfig) error

        // Restore tries to check the fault type and restore
        // the MySQL Cluster.
        Restore() error

        // CheckFaultType checks the fault type.
        CheckFaultType(*FaultInfo) (FaultType, error)

        // ProcessFault tries to process the fault.
        ProcessFault(FaultType, *FaultInfo) error

        // CheckStatus checks the status of the MySQL Cluster
        // periodically.
        CheckStatus() (*Status, error)

        // Feedback reports to the controller the status of the
        // MySQL Cluster.
        FeedBack(*Status) error
}
```

交付_Chart

- Chart 是 Kubernetes 生态圈中开发、分享、使用应用的标准
- 交付 Core Chart 和 Vendor Chart
- 通过 helm 来一键使用 Chart

交付_Chart 示例

```
➜ ~ helm create example
Creating example
➜ ~ tree example
example
|-- charts
|-- Chart.yaml
|-- templates
|   |-- deployment.yaml
|   |-- _helpers.tpl
|   |-- ingress.yaml
|   |-- NOTES.txt
|   `-- service.yaml
`-- values.yaml
```

交付_部署 MySQL Operator

```
➜ helm inspect ./mysql-operator
apiVersion: v1
…
---
replicaCount: 1
name: mysql-operator
registry: index-dev.qiniu.io/kelibrary
imageOperator: mysql-operator:201709281752
nameMySQLOperatorServer: mysql-operator-server
imageServer: mysql-operator-server:201709281752
…
# Install Default RBAC roles and bindings
rbac:
  install: false
```

交付_部署 MySQL Operator

```
➜ helm install --namespace rds --set name=example-mysql ./mysql-operator/
```

交付_部署 MySQL 服务

```
➜ helm inspect ./mysql-service
...
---
name: example
backup:
  enable: true
  backupTime: 01:01,13:01
  reservedDays: 7
resources:
  limits:
    cpu: 100m
    memory: 500Mi
    storage: 1Gi
uid: default
mysqlRootPassword: PASSWORD
```

交付_部署 MySQL 服务

```
➜ helm install --namespace rds --set name=example-mysql ./mysql-service/
```

总结

- 通过 Operator 和 Controller 封装 MySQL 领域内知识
- 简化 Operator 逻辑，通过 Controller 实现 MySQL 集群自治
- Controller 统一管理所在 MySQL 集群的任务分发
- 以 Chart 方式交付应用

Thanks