



## **RadonDB MySQL Kubernetes 2.1.3 用户文档**

RadonDB 开源社区

2022/4/7

# 1 关于 RadonDB MySQL Kubernetes

## 1.1 什么是 RadonDB MySQL Kubernetes?

RadonDB MySQL 是一款基于 MySQL 的开源、高可用、云原生集群解决方案。支持一主多从高可用架构，并具备安全、自动备份、监控告警、自动扩容等全套管理功能。目前已经在生产环境中大规模的使用，用户包含 银行、保险、传统大企业等。服务高可用由已经开源的 MySQL 集群高可用组建 [Xenon](#) 来实现。

随着国内外云原生技术蓬勃发展，数据库容器化实现技术趋于成熟，各类 K8s 社区用户对 MySQL on Kubernetes 高可用的需求呼声不断。社区决定将 RadonDB MySQL 完整的移植到 Kubernetes 平台，并于 2021 年将其正式开源。项目意在为广大的 Kubernetes 和 MySQL 开发者们，提供一款企业级的 MySQL on Kubernetes 高可用方案。

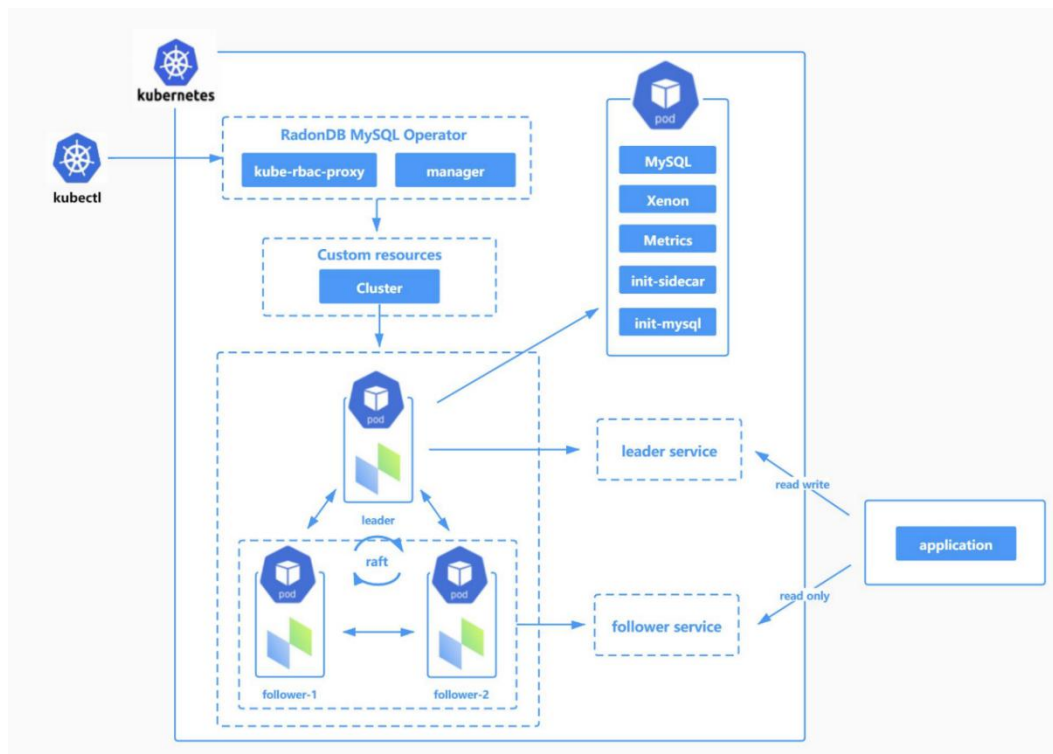
[RadonDB MySQL Kubernetes](#) 支持在 Kubernetes、KubeSphere、Rancher 等平台安装部署和管理，自动执行与运行 RadonDB MySQL 集群有关的任务。

### 1.1.1 核心功能

- MySQL 高可用
  - 无中心化自动选主
  - 主从秒级切换
  - 集群切换的数据强一致性
- 集群管理
- 监控告警
- 备份
- 集群日志管理
- 账户管理

### 1.1.2 架构图

- 通过 Raft 协议实现无中心化领导者自动选举
- 通过 Semi-Sync 基于 GTID 模式同步数据
- 通过 [Xenon](#) 提供高可用能力



### 1.1.3 Roadmap

1.0 Helm Chart	2.0 Operator	3.0 Operator
<ul style="list-style-type: none"> <li>● MySQL 高可用</li> <li>● 无中心化领导自动选举</li> <li>● 主从秒级切换</li> <li>● 数据强一致性</li> <li>● 集群管理</li> <li>● 监警告警</li> <li>● 集群日志管理</li> <li>● 账户管理</li> </ul>	<ul style="list-style-type: none"> <li>● 增删节点</li> <li>● 自动扩缩容</li> <li>● 升级集群</li> <li>● 备份与恢复</li> <li>● 故障自动转移</li> <li>● 自动重建节点</li> <li>● 自动重启服务</li> <li>● 账户管理 (提供 API 接口)</li> <li>● 在线迁移</li> </ul>	<ul style="list-style-type: none"> <li>● 自动化运维</li> <li>● 多节点角色</li> <li>● 灾备集群</li> <li>● SSL 传输加密</li> </ul>

### 1.1.4 协议

RadonDB MySQL 基于 Apache 2.0 协议，详见 [License](#)。

## 1.2 功能优势

- **数据强一致**

采用一主多备高可用架构，自动脑裂保护处理。

- **高可用**

支持一主多备架构，灵活满足各类可用性需求。

- **自动运维**

可设置自动备份策略、监控告警策略、自动扩容策略。

- **弹性扩缩容**

根据业务需要实时扩展数据库的 CPU、内存、存储容量。

## 1.3 应用场景

- **数据一致性较高的金融场景**

数据强一致性保证，满足金融级可靠性要求。

- **网站运维**

包含全套备份、恢复、监控等运维方案，满足网站类的业务需求。

- **高性能高并发场景**

满足高性能、高并发要求的业务，如电商秒杀等突发高峰场景。

- **弹性扩缩容分钟级部署**

对计算资源的弹性伸缩能力，分钟级部署游戏分区数据库。

## 2 部署

### 2.1 在 Kubernetes 上部署

RadonDB MySQL Kubernetes 2.0 版本通用。

本文档演示 RadonDB MySQL Kubernetes 在 Kubernetes 上的部署、校验、访问和卸载操作。

#### 2.1.1 部署准备

- Kubernetes 集群
- MySQL 客户端工具

#### 2.2.2 部署步骤

##### 2.2.2.1 添加 Helm 仓库

添加 Helm 仓库 radondb。

```
$ helm repo add radondb https://radondb.github.io/radondb-mysql-kubernetes/
```

校验仓库信息，可查看到名为 radondb/mysql-operator 的 chart。

```
$ helm search repo
```

NAME	CHART VERSION	APP VERSION	
DESCRIPTION			
radondb/mysql-operator	0.1.0	v2.1.x	Open
Source, High Availability Cluster, based on MySQL			

### 2.2.2.2 部署 Operator

设置 release 名为 demo , 创建名为 demo-mysql-operator 的 [Deployment](#)。

```
$ helm install demo radondb/mysql-operator
```

在这一步中，默认将同时创建集群所需的 [CRD](#)。

### 2.2.2.3 部署 RadonDB MySQL 集群

执行以下指令，以默认参数为 CRD mysqlclusters.mysql.radondb.com 创建一个实例，即创建 RadonDB MySQL 集群。

```
$ kubectl apply -f https://github.com/radondb/radondb-mysql-  
kubernetes/releases/latest/download/mysql_v1alpha1_mysqlcluster.yaml
```

自定义集群部署参数，可参考 [配置参数](#)。

## 2.2.3 部署校验

### 2.2.3.1 校验 RadonDB MySQL Operator

查看 demo 的 Deployment 和对应监控服务，回显如下信息则部署成功。

```
$ kubectl get deployment,svc
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
------	-------	------------	-----------	-----

demo-mysql-operator	1/1	1	1	7h50m
---------------------	-----	---	---	-------

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
------	------	------------	-------------	---------

AGE
-----

service/mysql-operator-metrics	ClusterIP	10.96.142.22	<none>	
--------------------------------	-----------	--------------	--------	--

8443/TCP	8h
----------	----

### 2.2.3.2 校验 RadonDB MySQL 集群

执行如下命令，将查看到如下 CRD。

```
$ kubectl get crd | grep mysql.radondb.com
```

backups.mysql.radondb.com	2021-11-02T07:00:01Z
mysqlclusters.mysql.radondb.com	2021-11-02T07:00:01Z
mysqlusers.mysql.radondb.com	2021-11-02T07:00:01Z

以默认部署为例，执行如下命令将查看到名为 sample-mysql 的三节点 RadonDB

MySQL 集群及用于访问节点的服务。

```
$ kubectl get statefulset,svc
```

NAME	READY	AGE
sample-mysql	3/3	7h33m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
service/sample-follower	ClusterIP	10.96.131.84	<none>	3306/TCP
7h37m				
service/sample-leader	ClusterIP	10.96.111.214	<none>	3306/TCP
7h37m				
service/sample-mysql	ClusterIP	None	<none>	3306/TCP
7h37m				

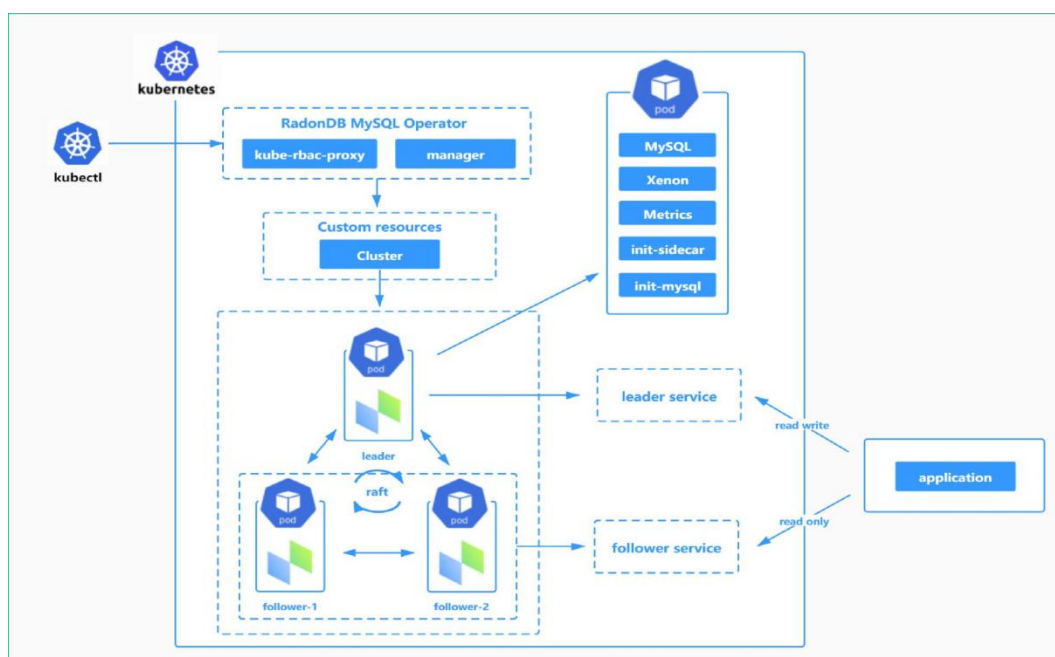


### 2.2.3.3 访问集群

在 Kubernetes 集群内，支持使用 service\_name 或者 clusterIP 方式，访问

RadonDB MySQL。

RadonDB MySQL 提供 Leader 和 Follower 两种服务，分别用于客户端访问主从节点。Leader 服务始终指向主节点（可读写），Follower 服务始终指向从节点（只读）。



以下为客户端与数据库在同一 Kubernetes 集群内，访问 RadonDB MySQL 的方式。

当客户端的与数据库部署在不同 Kubernetes 集群，请参考 [Kubernetes 访问集群中的应用程序](#)，配置端口转发、负载均衡等连接方式。

### 2.2.3.4 ClusterIP 方式

RadonDB MySQL 的高可用读写 IP 指向 Leader 服务的 clusterIP，高可用只读 IP 指向 Follower 服务的 clusterIP。

```
$ mysql -h <clusterIP> -P <mysql_Port> -u <user_name> -p
```

以下示例用户名为 radondb\_usr，Leader 服务的 clusterIP 为 10.10.128.136，连接示例如下：

```
$ mysql -h 10.10.128.136 -P 3306 -u radondb_usr -p
```

### 2.2.3.5 service\_name 方式

Kubernetes 集群的 Pod 之间支持通过 service\_name 方式访问 RadonDB MySQL。

service\_name 方式不适用于从 Kubernetes 集群的物理机访问数据库 Pod。

连接 Leader 服务（RadonDB MySQL 主节点）

```
$ mysql -h <leader_service_name>.<namespace> -u <user_name> -p
```

用户名为 radondb\_usr，release 名为 sample，RadonDB MySQL 命名空间为 default，连接示例如下：

```
$ mysql -h sample-leader.default -u radondb_usr -p
```

- 连接 Follower 服务（RadonDB MySQL 从节点）

```
$ mysql -h <follower_service_name>.<namespace> -u <user_name> -p
```

用户名为 radondb\_usr，release 名为 sample，RadonDB MySQL 命名空间为 default，连接示例如下：

```
$ mysql -h sample-follower.default -u radondb_usr -p
```

## 2.2.4 卸载

### 2.2.4.1 卸载 Operator

卸载当前命名空间下 release 名为 demo 的 RadonDB MySQL Operator。

```
$ helm delete demo
```

### 2.2.4.2 卸载集群

卸载 release 名为 sample RadonDB MySQL 集群。

```
$ kubectl delete mysqlclusters.mysql.radondb.com sample
```

### 2.2.4.3 卸载自定义资源

```
$ kubectl delete customresourcedefinitions.apiextensions.k8s.io
```

```
mysqlclusters.mysql.radondb.com
```

```
$ kubectl delete customresourcedefinitions.apiextensions.k8s.io
```

```
mysqlusers.mysql.radondb.com
```

```
$ kubectl delete customresourcedefinitions.apiextensions.k8s.io
```

```
backups.mysql.radondb.com
```

## 2.2 在 KubeSphere 上部署

RadonDB MySQL Kubernetes 2.0 版本通用。

本文档演示在 [KubeSphere](#) 上部署 RadonDB MySQL Kubernetes 的 Operator 和 MySQL 高可用集群。

### 2.2.1 部署准备

- 确保已启用 [OpenPitrix 系统](#)
- 创建一个企业空间、一个项目和一个用户供本 [操作使用](#)
  - 安装过程中，请以 admin 身份登录控制台，在企业空间 demo 中的 demo-project 项目中进行操作
- 确保 KubeSphere [项目网关](#) 已开启外网访问

### 2.2.2 部署步骤

#### 2.2.2.1 添加应用仓库

1. 登录 KubeSphere 的 Web 控制台。
2. 在 demo 企业空间中，进入应用管理下的应用仓库页面，点击添加，弹出仓库配置对话框。
3. 输入仓库名称和仓库 URL。
  - 输入 radondb-mysql-operator 作为应用仓库名称。
  - 输入 <https://radondb.github.io/radondb-mysql-kubernetes/> 作为仓库的 URL，并点击验证以验证 URL，在 URL 旁边呈现一个绿色的对号，验证通过后，点击确定继续。

4. 将仓库成功导入到 KubeSphere 之后，在列表中即可查看 RadonDB MySQL 仓库。

添加应用仓库

名称 \*

radondb-mysql-operator

URL

https://

▼

https://radondb.github.io/radondb-mysql-kubernetes

验证

URL 需要通过验证才能添加或编辑应用仓库。

同步周期 \*

0

秒

▼

设置一个同步周期，取值范围为 3 分钟到 24 小时。默认值 0 表示不同步。

描述

描述可包含任意字符，最长 256 个字符。

取消

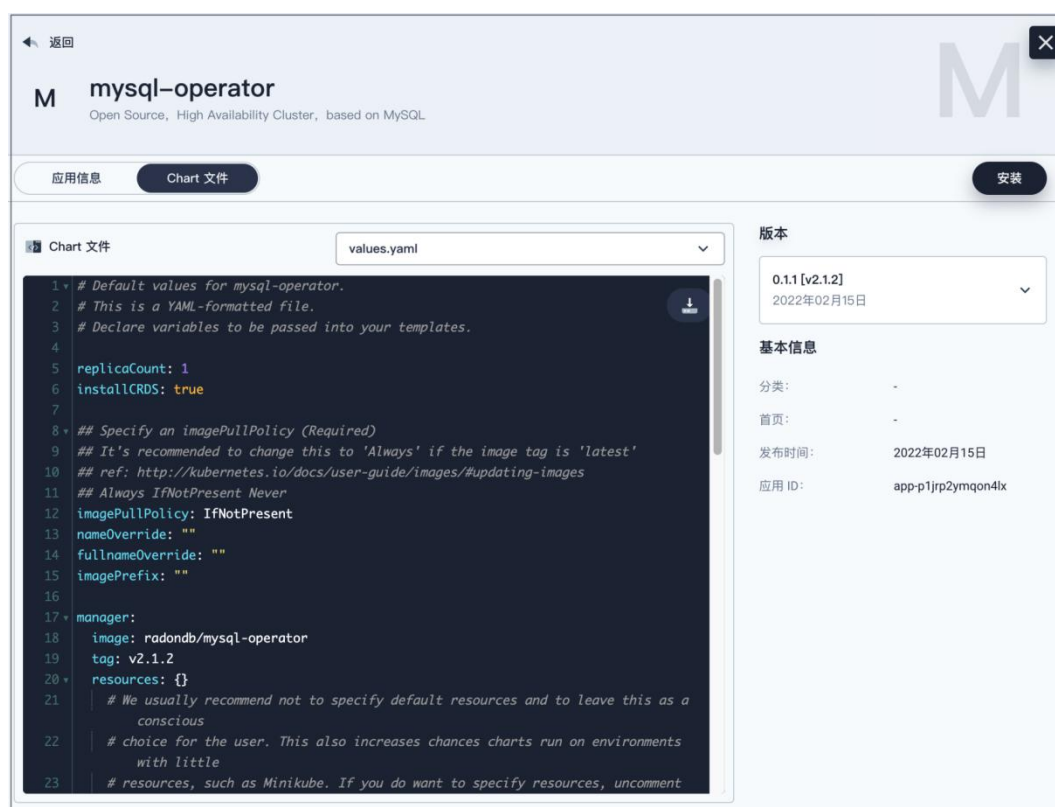
确定

### 2.2.2.2 部署 RadonDB MySQL Operator

1. 在 demo-project 项目中，进入应用负载下的应用页面，点击部署新应用。
2. 在对话框中，选择来自应用模板，进入应用模版页面。
3. 从下拉菜单中选择 radondb-mysql-operator 应用仓库。
4. 点击 mysql-operator 应用图标，查看和配置应用信息。

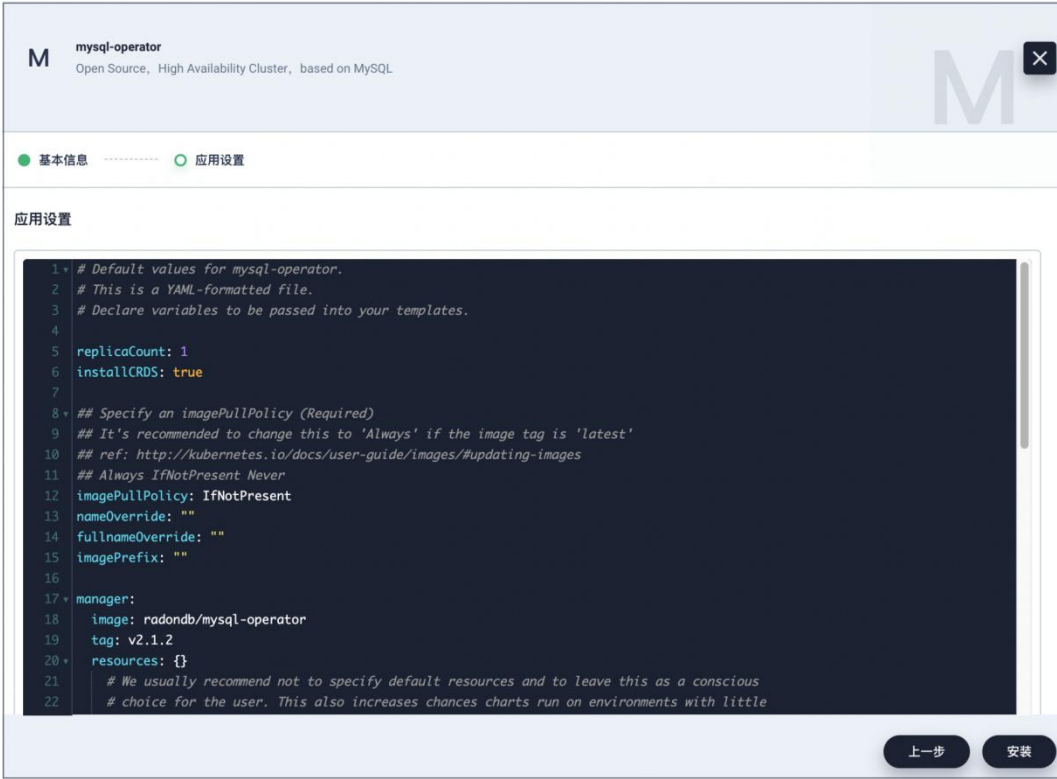


5. 在配置文件选项卡，可查看和编辑 values.yaml 配置文件；在版本列框区域，可查看和选择版本号。



6. 点击部署，进入 mysql-operator 应用基本信息配置页面，确认应用名称、应用版本以及配置部署位置。

7. 点击下一步，进入 mysql-operator 应用配置页面，确认 values.yaml 配置信息，可编辑文件修改配置。



8. 点击部署，返回应用模版页面。待应用状态切换为运行中，则应用部署成功。



### 2.2.2.3 更新 Operator

若已在 KubeSphere 部署过历史版本 Operator，可以选择如下方式更新到最新版本。

1. 在 KubeSphere 平台删除历史版本 Operator 应用。
2. 参考如上步骤，安装最新版本 Operator。
3. 执行如下命令更新 CRD 版本。如下示例为更新 CRD 到 2.1.2 版。

```
$ kubectl apply -f https://raw.githubusercontent.com/radondb/radondb-mysql-kubernetes/v2.1.2/charts/mysql-operator/crds/mysql.radondb.com_mysqlclusters.yaml
```

## 2.2.3 部署 RadonDB MySQL 集群

### 2.2.3.1 部署集群

可任选一个 RadonDB MySQL [配置示例](#) 部署，或自定义配置部署。

以 mysql\_v1alpha1\_mysqlcluster.yaml 模版为例，创建一个 RadonDB MySQL 集群。

1. 在右下角 工具箱中选择 Kubectl 工具，打开终端窗口。
2. 执行以下命令，安装 RadonDB MySQL 集群。

```
$ kubectl apply -f https://github.com/radondb/radondb-mysql-kubernetes/releases/latest/download/mysql_v1alpha1_mysqlcluster.yaml --namespace=<project_name>
```



## 注意

未指定项目时，集群将被默认安装在 `kubesphere-controls-system` 项目中。若需指定项目，安装命令需添加 `--namespace=<project_name>`。

## 预期结果

```
$ kubectl apply -f https://github.com/radondb/radondb-mysql-
kubernetes/releases/latest/download/mysql_v1alpha1_mysqlcluster.yaml --
namespace=demo-project
mysqlcluster.mysql.radondb.com/sample created
```

3. 集群创建成果后，执行如下命令，可查看 RadonDB MySQL 集群节点服务。

```
$ kubectl get statefulset,svc
```

## 预期结果

```
$ kubectl get statefulset,svc
```

NAME	READY	AGE
statefulset.apps/sample-mysql	3/3	10m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/default-http-backend	ClusterIP	10.96.69.202	<none>	80/TCP	3h2m
service/sample-follower	ClusterIP	10.96.9.162	<none>	3306/TCP	10m
service/sample-leader	ClusterIP	10.96.255.188	<none>	3306/TCP	

10m

service/sample-mysql      ClusterIP   None   <none>   3306/TCP

10m

2.2.3.2 部署校验

在 demo-project 项目中，查看 RadonDB MySQL 集群状态。

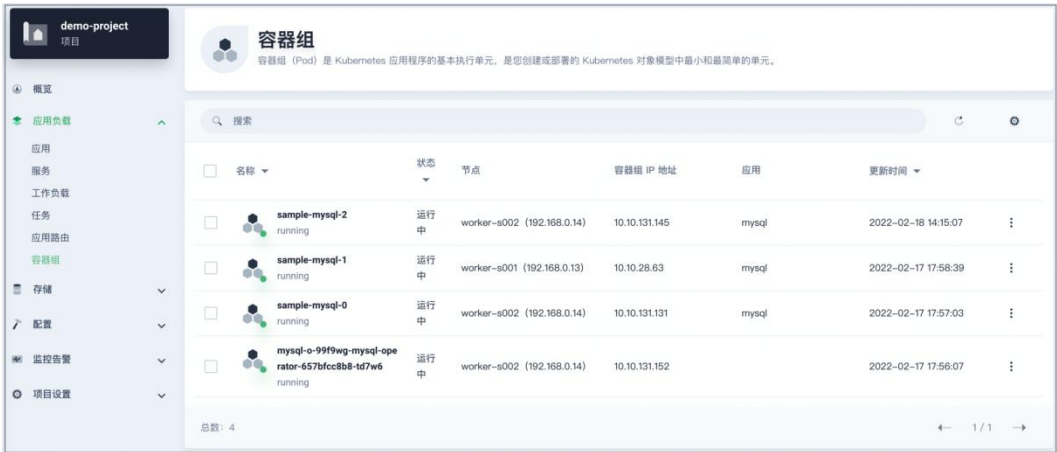
1. 进入 **应用负载** 下的 **服务** 页面，可查看集群服务信息。



2. 进入 **应用负载** 下的 **工作负载** 页面，点击 **有状态副本集**，可查看集群状态。进入一个 **有状态副本集** 详情页面，点击 **监控** 标签页，可查看一定时间范围内的集群指标。



3. 进入 应用负载 下的 容器组 页面，可查看集群节点运行状态。



4. 进入 存储 下的 存储卷 页面，可查看存储卷。查看某个存储卷用量信息，以其中一个数据节点为例，可以看到当前存储的存储容量和剩余容量等监控数据。



至此，完成在 KubeSphere 中部署 RadonDB MySQL 集群。

## 2.3 配置参数

### 2.3.1 容器配置

参数	描述	默认值
MysqlVersion	MySQL 版本号	5.7
MysqlOpts.RootPassword	MySQL Root 用户密码	""
MysqlOpts.User	默认新建的 MySQL 用户名称	radondb_usr
MysqlOpts.Password	默认新建的 MySQL 用户密码	RadonDB@123
MysqlOpts.Database	默认新建的 MySQL 数据库名称	radondb
MysqlOpts.InitTokuDB	是否启用 TokuDB	true
MysqlOpts.MysqlConf	MySQL 配置	-
MysqlOpts.Resources	MySQL 容器配额	预留: cpu 100m, 内存 256Mi; 限制: cpu 500m, 内存 1Gi
XenonOpts.Image	xenon(高可用组件) 镜像	radondb/xenon:1.1.5-alpha
XenonOpts.AdmitDefeatHeartbeatCount	允许的最大心跳检测失败次数	5
XenonOpts.ElectionTimeout	选举超时时间(单位为毫秒)	10000ms
XenonOpts.Resources	xenon 容器配额	预留: cpu 50m, 内存 128Mi; 限制: cpu 100m, 内存 256Mi
MetricsOpts.Enabled	是否启用 Metrics(监控)容器	false
MetricsOpts.Image	Metrics 容器镜像	prom/mysqld-exporter:v0.12.1
MetricsOpts.Resources	Metrics 容器配额	预留: cpu 10m, 内存 32Mi;

		限制: cpu 100m, 内存 128Mi
--	--	------------------------

### 2.3.2 节点配置

参数	描述	默认值
Replicas	集群节点数, 只允许为 0、2、3、5	3
PodPolicy.ImagePullPolicy	镜像拉取策略, 只允许为 Always/IfNotPresent/Never	IfNotPresent
PodPolicy.Labels	节点 pod <a href="#">标签</a>	-
PodPolicy.Annotations	节点 pod <a href="#">注解</a>	-
PodPolicy.Affinity	节点 pod <a href="#">亲和性</a>	-
PodPolicy.PriorityClassName	节点 pod <a href="#">优先级</a> 对象名称	-
PodPolicy.Tolerations	节点 pod <a href="#">污点容忍度</a> 列表	-
PodPolicy.SchedulerName	节点 pod <a href="#">调度器</a> 名称	-
PodPolicy.ExtraResources	节点容器配额 (除 MySQL 和 Xenon 之外的容器)	预留: cpu 10m 内存 32Mi
PodPolicy.SidecarImage	Sidecar 镜像	radondb/mysql-sidecar:latest
PodPolicy.BusyboxImage	Busybox 镜像	busybox:1.32
PodPolicy.SlowLogTail	是否开启慢日志跟踪	false
PodPolicy.AuditLogTail	是否开启审计日志跟踪	false

### 2.3.3 持久化配置

参数	描述	默认值
Persistence.Enabled	是否启用持久化	true
Persistence.AccessModes	存储卷访问模式	ReadWriteOnce
Persistence.StorageClasses	存储卷类型	-
Persistence.Size	存储卷容量	10Gi

### 2.3.4 配置示例

apiVersion: mysql.radondb.com/v1alpha1

kind: MysqlCluster

metadata:

name: sample

spec:

replicas: 3

mysqlVersion: "5.7"

# the backupSecretName specify the secret file name which store S3

information,

# if you want S3 backup or restore, please create backup\_secret.yaml,

uncomment below and fill secret name:

# backupSecretName:

# if you want create mysqlcluster from S3, uncomment and fill the directory

in S3 bucket below:

# restoreFrom:

mysqlOpts:

rootPassword: "RadonDB@123"

rootHost: localhost

user: radondb\_usr

password: RadonDB@123

database: radondb

initTokuDB: true

# A simple map between string and string.

# Such as:

# mysqlConf:

# expire\_logs\_days: "7"

mysqlConf: {}

resources:

requests:

cpu: 100m

memory: 256Mi

limits:

cpu: 500m

memory: 1Gi

xenonOpts:

image: radondb/xenon:1.1.5-alpha

admitDefeatHeartbeatCount: 5

electionTimeout: 10000

resources:

requests:

cpu: 50m

memory: 128Mi

limits:

cpu: 100m

memory: 256Mi

metricsOpts:

enabled: false

image: prom/mysql-d-exporter:v0.12.1

resources:

requests:

cpu: 10m

memory: 32Mi

limits:

cpu: 100m

memory: 128Mi

podPolicy:

imagePullPolicy: IfNotPresent

sidecarImage: radondb/mysql-sidecar:latest

busyboxImage: busybox:1.32

slowLogTail: false

auditLogTail: false

labels: {}

annotations: {}



affinity: {}

priorityClassName: ""

tolerations: []

schedulerName: ""

# extraResources defines quotas for containers other than mysql or xenon.

extraResources:

requests:

cpu: 10m

memory: 32Mi

persistence:

enabled: true

accessModes:

- ReadWriteOnce

#storageClass: ""

size: 20Gi

## 3 功能介绍

### 3.1 监控与告警

RadonDB MySQL Kubernetes 2.1.0+ 支持。

#### 3.1.1 背景

[Prometheus](#) 基于文本的暴露格式，已经成为云原生监控领域事实上的标准格式。

RadonDB MySQL 监控引擎基于 [Prometheus MySQLd Exporter](#) 定义。通过 mysql-d-exporter 抓取 RadonDB MySQL 服务指标，再通过接入第三方应用平台实现监控指标可视化。

本教程演示如何开启 RadonDB MySQL 监控指标。

#### 3.1.2 准备工作

- 已准备可用 [Kubernetes](#) 或 [KubeSphere](#) 集群

#### 3.1.3 部署步骤

##### 3.1.3.1 配置 serviceMonitor

serviceMonitor 是定义 RadonDB MySQL Operator 自动监控引擎的参数，开启后

将自动绑定 `mysqld_exporter` 与 Prometheus。

`serviceMonitor` 参数包含如下字段：

`serviceMonitor:`

`enabled: true`

`## Additional labels for the serviceMonitor. Useful if you have multiple`

`prometheus operators running to select only specific ServiceMonitors`

`# additionalLabels:`

`# prometheus: prom-internal`

`interval: 10s`

`scrapeTimeout: 3s`

`# jobLabel:`

`# targetLabels:`

`# podTargetLabels:`

`namespaceSelector:`

`any: true`

`selector:`

`matchLabels:`

`app.kubernetes.io/managed-by: mysql.radondb.com`

`app.kubernetes.io/name: mysql`

您可以在 `charts/mysql-operator/values.yaml` 文件中配置 `serviceMonitor`。

- 新部署 Operator 时，`serviceMonitor.enabled` 默认为 **true**，表示默认开启。

- 已部署 Operator 2.1.0 以下版本的集群，需重新部署 Operator。

### 3.1.3.2 配置 metricsOpts

metricsOpts 是 CRD `mysqlclusters.mysql.radondb.com` 中定义 RadonDB MySQL 集群监控的参数，可通过配置 `mysql_v1alpha1_mysqlcluster.yaml` 文件中参数值开启监控服务。

metricsOpts 参数包含如下字段：

metricsOpts:

enabled: false

image: prom/mysqld-exporter:v0.12.1

resources:

requests:

cpu: 10m

memory: 32Mi

limits:

cpu: 100m

memory: 128Mi

metricsOpts.enabled 默认为 **false**，需手动设置为 **true**。

- 选择设置 metricsOpts.enabled 状态为 **true**，开启集群监控功能。

- 设置资源参数值，定义监控容器资源配额大小。

文件参数修改完成后，使用如下指令应用配置，部署/更新集群回显信息如下：

```
$ kubectl apply -f config/sample/mysql_v1alpha1_mysqlcluster.yaml  
cluster.mysql.radondb.com/sample created/configured
```

### 3.1.4 查看监控服务

#### 3.1.4.1 通过客户端查看

您可以通过如下指令查看集群监控服务和 serviceMonitor 信息。

```
$ kubectl get service,servicemonitor  
  
$ kubectl describe servicemonitor <serviceName>
```

#### 预期效果

```
$ kubectl get service,servicemonitor
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
service/mysql-operator-metrics	ClusterIP	10.96.242.205	<none>	8443/TCP
service/sample-follower	ClusterIP	10.96.2.234	<none>	3306/TCP
service/sample-leader	ClusterIP	10.96.30.238	<none>	3306/TCP

21h

service/sample-metrics	ClusterIP	10.96.7.222	<none>	9104/TCP
------------------------	-----------	-------------	--------	----------

3h24m

service/sample-mysql	ClusterIP	None	<none>	3306/TCP
----------------------	-----------	------	--------	----------

21h

NAME

AGE

servicemonitor.monitoring.coreos.com/demo-mysql-operator	3h25m
--	-------

\$ kubectl describe servicemonitor demo-mysql-operator

Name: test-radondb-mysql-metrics

Namespace: default

Labels: app=test-radondb-mysql

app.kubernetes.io/managed-by=Helm

app.kubernetes.io/vendor=kubesphere

chart=radondb-mysql-1.0.0

heritage=Helm

release=test

Annotations: kubesphere.io/creator: admin

API Version: monitoring.coreos.com/v1

Kind: ServiceMonitor

.....

Spec:

Endpoints:

Interval: 1m

Path: /metrics

Port: metrics

Scheme: http

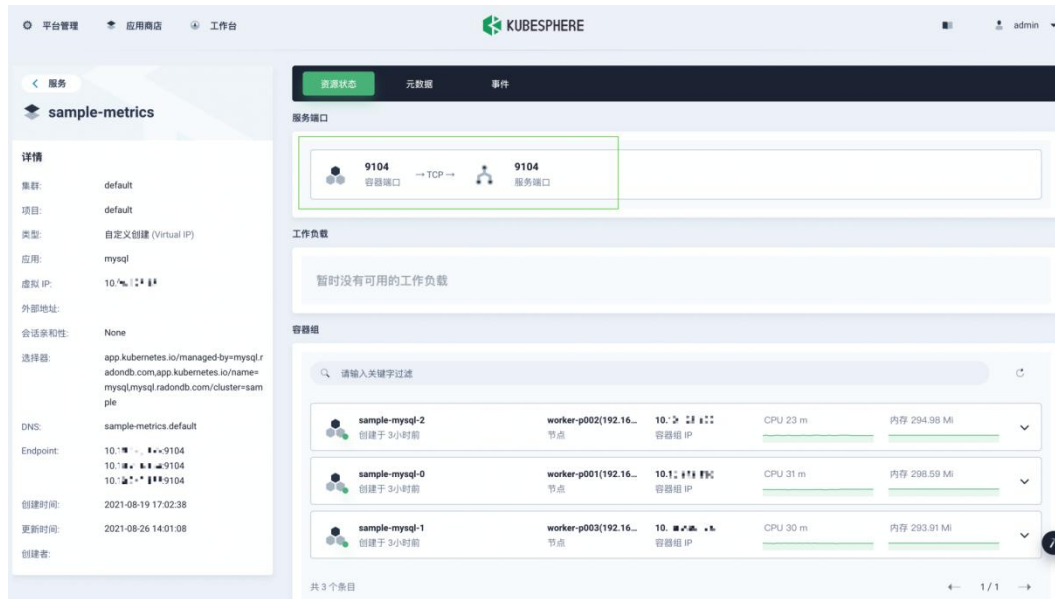
Scrape Timeout: 10s

.....

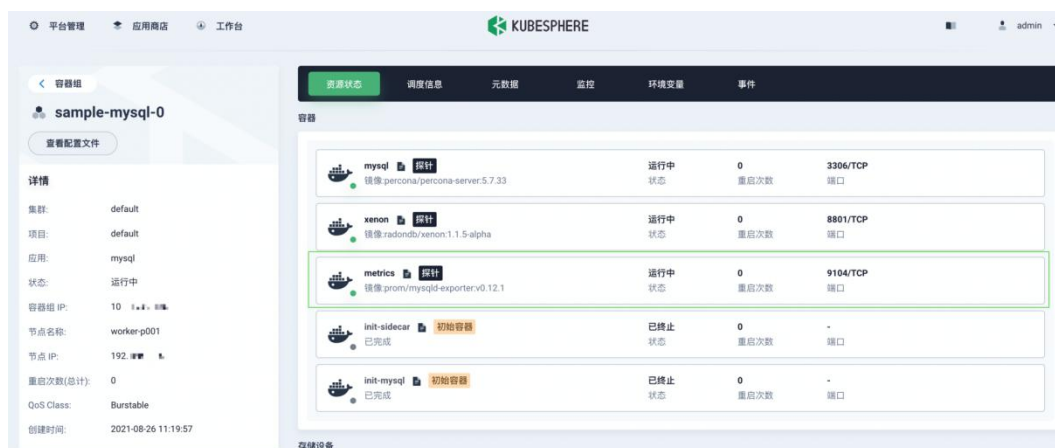
### 3.1.4.2 在 KubeSphere 平台查看

在 KubeSphere 企业空间部署的 RadonDB MySQL Operator 和 MySQL 集群，开启监控后，可在如下页面查看监控服务状态。

- 在项目空间 **应用负载** 下的 **服务** 页面，点击 <集群名称>-metrics，可查看监控服务信息。



- 在项目空间 **应用负载** 下的 **容器组** 页面，点击一个容器的名称，可查看该容器中 metrics 资源状态。



### 3.1.5 查看监控

#### 3.1.5.1 通过 KubeSphere 自定义监控

RadonDB MySQL Operator 和集群需部署在 KubeSphere。

KubeSphere 的监控引擎基于 Prometheus 和 Prometheus Operator。使用 KubeSphere 的自定义监控功能支持以可视化的形式监控 RadonDB MySQL 指标。

1. 在集群同一项目中，选择**监报告警**下的**自定义监控**，点击**创建**。



## 2. 在对话框中，选择 **MySQL** 模版，并继续配置监控模版。

创建自定义监控面板

编辑模式

×

名称 \*


最长 63 个字符，只能包含小写字母、数字及分隔符("-")，且必须以小写字母或数字开头及结尾

描述信息

描述信息不超过 256 个字符


选择适合您应用应用模板

监控面板将根据应用类型生成默认的面板配置

  
**Elasticsearch**  
elasticsearch-exporter

  
**Mysql**  
Prometheus Mysql Exporter

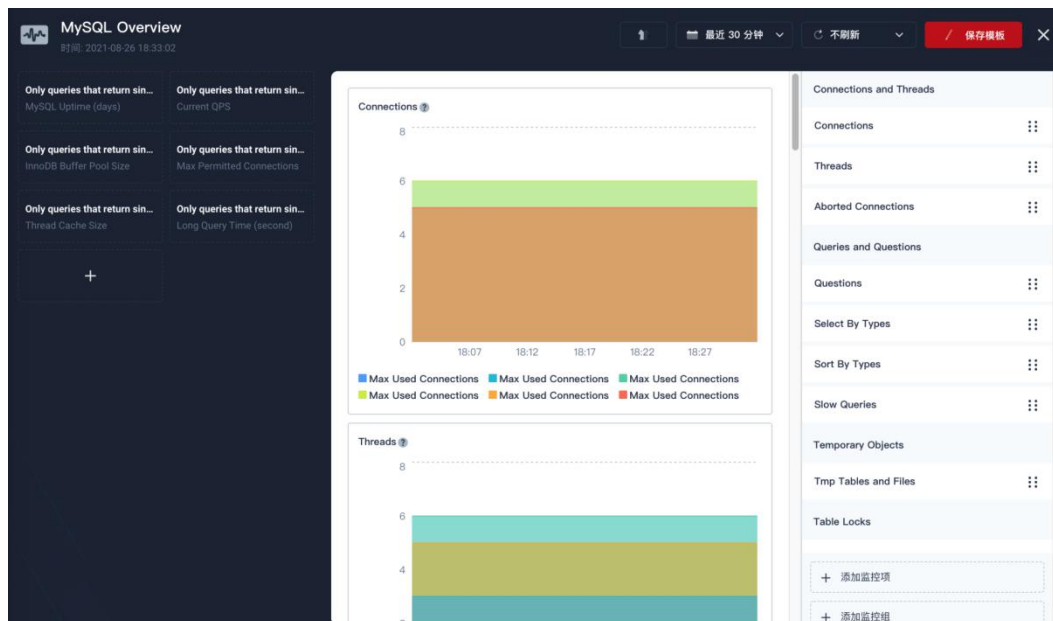
  
**Redis**  
Prometheus Redis Exporter

  
**Custom**  
Customize Your Exporter

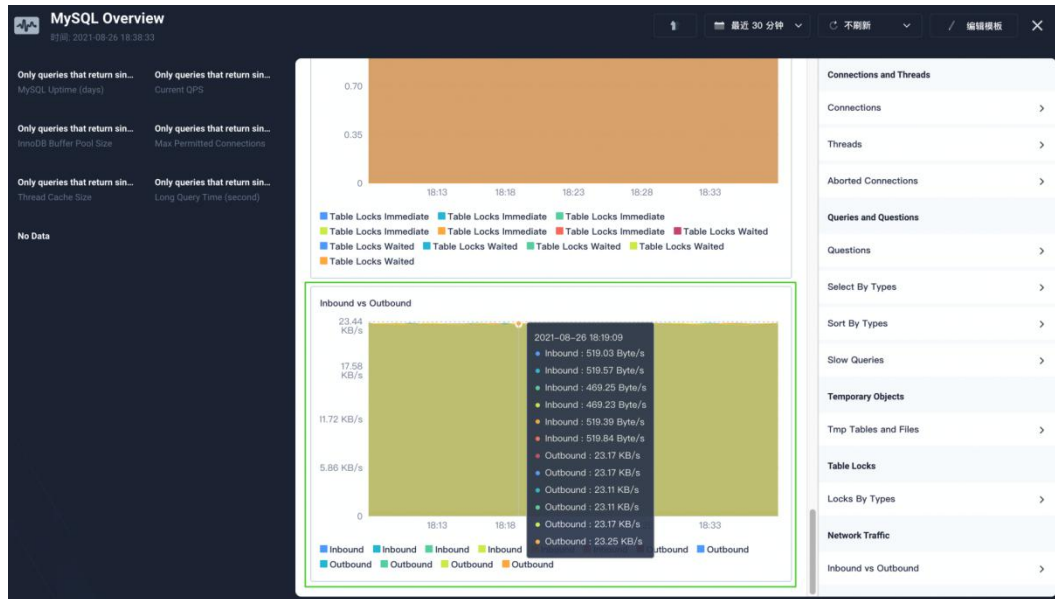
取消

下一步

## 3. 点击**保存模版**，即新创建监控面板。



## 4. 新建监控面板需等待约十分钟，即可查看监控数据。

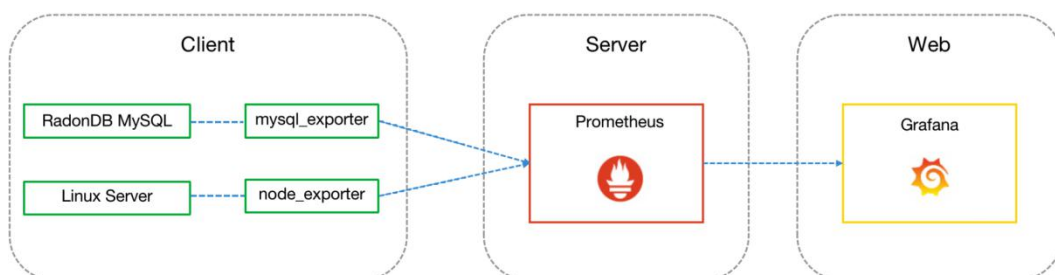


更多详情，请查看 KubeSphere [自定义监控介绍](#) 和 [可视化监控](#)。

### 3.1.5.2 通过 Prometheus + Grafana 平台

[Grafana](#) 是一个跨平台、开源的数据可视化网络应用程序平台。通过 Prometheus + Grafana 平台查看监控基本原理如下：

- 通过 [mysql\\_exporter](#) 获取 RadonDB MySQL 服务监控数据。
- 通过 [node\\_exporter](#) 获得 RadonDB MySQL 服务器的监控数据。
- 将监控数据传到 [Prometheus](#) 后，通过配置数据源，最终在 Grafana 呈现丰富的监控数据图表和警告。



更多 Grafana 可视化监控使用说明，请参见 [Grafana Dashboards](#)。

## 3.2 备份与恢复

RadonDB MySQL Kubernetes 2.1.0+ 支持。

### 3.2.1 快速开始备份

#### 3.2.1.1 安装 Operator

安装名为 test 的 Operator。

```
$ helm install test charts/mysql-operator
```

#### 3.2.1.2 配置备份

添加保密文件。

kind: Secret

apiVersion: v1

metadata:

name: sample-backup-secret

namespace: default

data:

s3-endpoint: aHR0cDovL3MzLnNoMWEucWluZ3N0b3luY29t

s3-access-key: SEdKWldXVIIlSENISIIlFRERKSUc=

s3-secret-key:

TU44TkNUdDJldHlZREROTTc5cTNwdkxtNTl0E01blRaZlRQMWxoag==

s3-bucket: bGFsYS1teXNxbA==

type: Opaque

s3-xxxx 的值是试用 base64 算法加密的，你可以这样获得。

```
$ echo -n "hello"|base64
```

然后在 Kubernetes 中创建加密配置。

```
$ kubectl create -f config/samples/backup_secret.yaml
```

请在 mysql\_v1a1pha1\_mysqlcluster.yaml 文件中添加 backupSecretName 属性。

spec:

replicas: 3

mysqlVersion: "5.7"

backupSecretName: sample-backup-secret

...

现在，如下创建备份文件 `mysql_v1a1pha1_backup.yaml` 如下。

```
apiVersion: mysql.radondb.com/v1alpha1
```

```
kind: Backup
```

```
metadata:
```

```
  name: backup-sample1
```

```
spec:
```

```
  # Add fields here
```

```
  hostname: sample-mysql-0
```

```
  clustname: sample
```

name	function
hostname	pod name in cluser
clustname	cluster name

### 3.2.1.3 开启集群

```
$ kubectl apply -f config/samples/mysql_v1alpha1_mysqlcluster.yaml
```

### 3.2.1.4 开始备份

在集群运行成功后开始备份。

```
$ kubectl apply -f config/samples/mysql_v1alpha1_backup.yaml
```

## 3.2.2 卸载

### 3.2.2.1 卸载 Operator

卸载名为 test 的 Operator。

```
$helm uninstall test
```

```
$kubectl delete -f config/samples/mysql_v1alpha1_backup.yaml
```

### 3.2.2.2 卸载集群

卸载名为 sample 的集群。

```
$ kubectl delete mysqlclusters.mysql.radondb.com sample
```

### 3.2.2.3 卸载资源

```
$ kubectl delete customresourcedefinitions.apiextensions.k8s.io
```

```
mysqlclusters.mysql.radondb.com
```

## 3.2.3 从备份中恢复集群

检测你的 S3 bucket，获取你备份的目录，比如：backup\_2021720827，并且将设置为 yaml 文件的 restoreFrom 属性中。

...

spec:

replicas: 3

mysqlVersion: "5.7"

```
backupSecretName: sample-backup-secret
```

```
restoreFrom: "backup_2021720827"
```

...

此时执行：

```
$ kubectl apply -f config/samples/mysql_v1alpha1_mysqlcluster.yaml
```

could restore a cluster from the backup\_2021720827 copy in the S3 bucket.

完成，已经从名为 backup\_2021720827 的 S3 备份中恢复一个集群。

### 3.2.3.1 创建镜像

如下：

```
$ docker build -f Dockerfile.sidecar -t acekingke/sidecar:0.1 . && docker push
```

```
acekingke/sidecar:0.1
```

```
$ docker build -t acekingke/controller:0.1 . && docker push
```

```
acekingke/controller:0.1
```

可以将 acekingke/sidecar:0.1 改为你自己的标签。

### 3.2.3.2 deploy your own manager

```
$ make manifests
```

```
$ make install
```

```
$ make deploy IMG=acekingke/controller:0.1 KUSTOMIZE=~/.radondb-mysql-
```

kubernetes/bin/kustomize

## 3.3 用户管理

RadonDB MySQL Kubernetes 2.1.0+ 支持。

### 3.3.1 准备工作

- 已部署 [RadonDB MySQL 集群](#)。

### 3.3.2 创建用户帐号

#### 3.3.2.1 校验 CRD

使用如下指令，将查看到名称为 mysqlusers.mysql.radondb.com 的 CRD。

```
$ kubectl get crd | grep mysqluser
```

```
mysqlusers.mysql.radondb.com          2021-09-21T09:15:08Z
```

#### 3.3.2.2 创建 Secret

RadonDB MySQL 使用 Kubernetes 中的 [Secret](#) 对象保存用户的密码。

使用如下指令，将使用 [示例配置](#) 创建一个名为 sample-user-password 的 Secret。

```
$ kubectl apply -f https://raw.githubusercontent.com/radondb/radondb-  
mysql-kubernetes/main/config/samples/mysqluser_secret.yaml
```



### 3.3.2.3 创建用户

使用如下指令，将使用 [示例配置](#) 创建一个名为 sample\_user 的用户。

```
$ kubectl apply -f https://raw.githubusercontent.com/radondb/radondb-mysql-kubernetes/main/config/samples/mysql_v1alpha1_mysqluser.yaml
```

注意：直接修改 spec.user（用户名）等同于以新用户名创建一个用户。如需创建多个用户，请确保 metadata.name（CR 实例名）与 spec.user（用户名）一一对应。

### 3.3.3 修改用户帐号

用户帐号信息由 spec 字段中参数定义，目前支持：

- 修改 hosts 参数。
- 新增 permissions 参数。

#### 3.3.3.1 授权 IP

允许使用用户帐号的 IP，通过 hosts 字段参数定义。

- % 表示所有 IP 均可访问。
- 可修改一个或多个 IP。

hosts:

- "%"

#### 3.3.3.2 用户权限

用户帐号数据库访问权限，通过 MysqlUser 中 permissions 字段参数定义。可通过新增 permissions 字段参数值，实现用户帐号权限的新增。

permissions:

- database: "\*"

tables:

- "\*"

privileges:

- SELECT

- database 参数表示该用户帐号允许访问的数据库。\* 代表允许访问集群所有数据库。
- tables 参数表示该用户帐号允许访问的数据库表。\* 代表允许访问数据库中所有表。
- privileges 参数表示该用户帐号被授权的数据库权限。更多权限说明，请参见 [Privileges Supported by MySQL](#)。

### 3.3.4 删除用户帐号

使用如下指令将删除使用 [示例配置](#) 创建的 MysqlUser CR。

```
$ kubectl delete mysqluser sample-user-cr
```

说明：删除 MysqlUser CR 会自动删除 CR 对应的 MySQL 用户。

### 3.3.5 示例配置

#### 3.3.5.1 Secret

apiVersion: v1

kind: Secret

metadata:

name: sample-user-password # 密钥名称。应用于 MysqlUser 中的

secretSelector.secretName。

data:

pwdForSample: UmFkb25EQkAxMjMKIA== #密钥键，应用于 MysqlUser 中的

secretSelector.secretKey。示例密码为 base64 加密的 RadonDB@123

# pwdForSample2:

# pwdForSample3:

### 3.3.5.2 MysqlUser

apiVersion: mysql.radondb.com/v1alpha1

kind: MysqlUser

metadata:

name: sample-user-cr # 用户 CR 名称，建议使用一个用户 CR 管理一个用户。

spec:

user: sample\_user # 需要创建/更新的用户的名称。

hosts: # 支持访问的主机，可以填多个，% 代表所有主机。

- "%"

permissions:

- database: "\*" # 数据库名称, \* 代表所有数据库。

tables: # 表名称, \* 代表所有表。

- "\*"

privileges: # 权限, 参考

<https://dev.mysql.com/doc/refman/5.7/en/grant.html>。

- SELECT

userOwner: # 指定被操作用户所在的集群。不支持修改。

clusterName: sample

nameSpace: default # radondb mysql 集群所在的命名空间。

secretSelector: # 指定用户的密钥和保存当前用户密码的键。

secretName: sample-user-password # 密钥名称。

secretKey: pwdForSample # 密钥键, 一个密钥可以保存多个用户的密码, 以键区分。

## 4 版本动态

### 版本发布汇总

#### 版本须知【重要】

- 1.x 版本由 Helm 包管理工具部署, 目前已停止维护!

- 2.x 版本由 Operator 的方式实现并兼容 1.x 所有功能。
- 强烈建议使用 2.x 最新版本！

## 近期 Roadmap

1. 支持更多方式的数据库备份恢复
2. 支持更细粒度的配置更新
3. 支持 MySQL 8.0
4. 抽象完善外部调用 API
5. 进一步提升服务质量，减少特殊场景下启停时间
6. 完善周期调度 job 功能更高效支持重复工作
7. 支持在线迁移
8. 完善 e2e 测试框架，覆盖更多场景

## RadonDB MySQL Kubernetes 版本

版本	发布时间
<a href="#">2.1.3</a>	2022-03-24
<a href="#">2.1.2</a>	2022-02-10
<a href="#">2.1.1-alpha</a>	2021-12-02
<a href="#">2.1.0</a>	2021-10-26
<a href="#">2.0.0</a>	2021-08-10
<a href="#">1.2.0</a>	2021-06-09
<a href="#">1.1.0</a>	2021-05-07
<a href="#">1.0.0</a>	2021-04-27

了解最新版本发布信息: <https://github.com/radondb/radondb-mysql-kubernetes/releases>

### **2.1.3**

RadonDB MySQL Kubernetes 于 3 月 24 日发布版本 2.1.3。该版本主要基于在 2.1.2 进行功能优化和升级。

### **致谢**

首先感谢 @andyli029 @acekingke @runkecheng @mgw2168 @molliezhong 提

交的修改。

## 新版本功能一览

1. 一键发布 workflow
2. 支持按标签重建集群节点
3. 增加 Pod 调试模式

以下是完整的 Release Notes。

### 2.1.3 Release Notes

#### Features

- workflow: Publish release only one click. [#421](#) ([#422](#))
- mysqlcluster: Support automatic rebuild of nodes by label. ([#389](#))
- mysqlcluster: Debug Mode for Pod [#375](#) ([#383](#))

#### Improvements

- .github: Adjust release-drafter ([#424](#))
- chart: Update chart version to v2.1.3. ([#419](#))
- config: Add podAntiAffinity sample yaml. [#371](#) ([#393](#))
- docs: Add troubleshoot.md [#387](#) ([#414](#))
- docs: Add offline deployment document. [#396](#) ([#399](#))
- docs: Add a description of service\_name connection method [#401](#) ([#402](#))

#### Bug Fixes

- cmd: Change HttpServer stop channel to buffered channel. [#411](#) ([#411](#))
- status: Skip the unavailable node and set default node status. [#417](#) ([#418](#))
- container: Add xenoncli check in the liveness probe. ([#405](#))
- syncer: Uniform use of global variables set role labels. ([#394](#))
- hack: Change Xenon's Dockerfile image branch to master. [#336](#) ([#392](#))

## 2.1.2

RadonDB MySQL on Kubernetes 于 2 月 17 日发布版本 2.1.2 。该版本在节点的重建、增删等方面进行了全面升级。

## 致谢

首先感谢 @andyli029 @acekingke @runkecheng @molliezhong 提交的修改。



## 新版本功能一览

1. 支持从已有节点克隆数据初始化
2. 支持重建节点
3. 支持显示节点 Raft 状态
4. 增删节点不再触发滚动更新
5. 支持一键配置镜像地址前缀
6. 增加多平台部署文档
7. 支持 e2e 测试框架

以下是完整的 Release Notes。

### 2.1.2 Release Notes

#### Features

- Clone init from follower node. [#322](#)
- Support for manual repair invalid nodes. [#331](#)
- Add E2E framework and simple testcase. [#347](#)
- Support more node role labels. [#334](#)
- Support unified setting images repository address. [#378](#)
- Add tutorials of deploy radondb mysql on rancher. [#338](#)
- Add tutorials of deploy radondb mysql on kubesphere. [#152](#)

#### Improvements

- Upgrade E2E frame to Ginkgo v2. [#360](#)

- Update the description about access radondb mysql. [#340](#)
- Change the default path of the rbac proxy image. [#146](#)
- Make the versions provided by helm repo and release consistent.  
[#352]<https://github.com/radondb/radondb-mysql-kubernetes/issues/352>
- Add .gitignore about e2e logs and function. [#381](#)

## Bug fixes

- Fixed the cluster status cannot be changed after the POD exit abnormally.  
[#366](#)
- Fixed the container time zone is not consistent with the host time zone .  
[#329](#)

## What's Changed

Full Changelog: [v2.0.0...v2.1.2](#)

## 2.1.1

RadonDB MySQL on Kubernetes 于 2021 年 12 月 2 日发布版本 2.1.1。

## 致谢

首先感谢 @andyli029 @acekingke @runkecheng @molliezhang 提交的修改。

以下是完整的 Release Notes。

## 2.1.1 Release Notes

### Features

- Support clone initial when add new pod. [#250#291](#)
- Update replicas without restart. [#282](#)
- Support display the raft status of the node in nodes.conditions. [#284#285](#)
- charts: Support offline deployment. [#300#301](#)
- workflow: Manage Chart using Helm repo. [#290#294](#)
- workflow: Automatic code check and unit tests. [#277](#)
- Makefile: Synchronize the generated files to Chart while generating CRD.  
[#280](#)

### Improvements

- syncer: Make Nodes.Conditions only show the condition of the presence node. [#283#286](#)
- syncer: Keep PVC when closing the cluster. [#304#308](#)
- syncer: Optimize update POD trigger conditions. [#321](#)
- sidecar: Rewrite restore logic using golang. [#292#293](#)
- container: Optimize the directive of Mysql liveness check. [#305#318](#)
- Dockerfile: Provide backup of district/static:nonroot image. [#287#296](#)
- docs: Update deployment document. [#298](#)

### Bug fixes

- Fix the setting method of innodb\_buffer\_pool\_instance. [#244#265](#)
- Fix bug of not effective version of mysql56. [#203#217](#)
- Fix failed to restore from backup after extending pvc. [#370#291](#)
- syncer: Fix bug of parallel updated nodes. [#310#314](#)
- syncer: Fix operator restart when closing cluster. [#312#315](#)
- container: Fix pod exception restart when high pressure. [#305#318](#)
- docs: Fix check CRD about mysqluser. [#281](#)

## 2.1.0

RadonDB MySQL Kubernetes 于 2021 年 10 月 22 日发布第四个版本 2.1.0。该版本也是由 Operator 方式实现的第二个版本。

## 致谢

首先感谢 @andyli029 @hustjieke @zhyass @runkecheng @acekingke

@molliezhang 提交的修改。

## 新版本功能一览

### 1. 增加 MySQL 集群服务监控

开启监控功能后，将创建监控服务并自动对接 Prometheus。

### 2. 基于 S3 的数据库备份恢复

只要拥有 S3 对象存储的 bucket 与 API key，直接将 Pod 的数据库内容备份到 S3 对象存储中，也可以从 S3 对象存储中的备份恢复出新的数据库集群。

### 3. 完善数据库账户管理

通过 CR 管理 MySQL 用户。对 CR 的增删改将自动转化为对相应用户的操作，支持针对数据库，表授权。

### 4. 支持磁盘动态扩容

可修改 yaml 存储容量，自动升级扩容存储，并升级数据库集群。

### 5. 优化优雅启停逻辑

### 6. 丰富集群状态粒度

支持集群中间状态显示，例如：初始化中，更新中等；新增集群状态已关闭。

### 7. 支持外网服务访问

### 8. 优化代码和迭代更新

### 9. 完善单元测试

### 10. 丰富工作流和 Travis CI 支持自动构建镜像，格式检查，单元测试

以下是完整的 Release Notes。

## 2.1.2 Release Notes

## Features

- Clone init from follower node. [#322](#)
- Support for manual repair invalid nodes. [#331](#)
- Add E2E framework and simple testcase. [#347](#)
- Support more node role labels. [#334](#)
- Support unified setting images repository address. [#378](#)
- Add tutorials of deploy radondb mysql on rancher. [#338](#)
- Add tutorials of deploy radondb mysql on kubesphere. [#152](#)

## Improvements

- Upgrade E2E frame to Ginkgo v2. [#360](#)
- Update the description about access radondb mysql. [#340](#)
- Change the default path of the rbac proxy image. [#146](#)
- Make the versions provided by helm repo and release consistent.  
[[#352](#)]<https://github.com/radondb/radondb-mysql-kubernetes/issues/352>)
- Add .gitignore about e2e logs and function. [#381](#)

## Bug fixes

- Fixed the cluster status cannot be changed after the POD exit abnormally.  
[#366](#)
- Fixed the container time zone is not consistent with the host time zone .  
[#329](#)

## What's Changed

Full Changelog: [v2.0.0...v2.1.2](#)

## 2.0.0

RadonDB MySQL on Kubernetes 于 2021 年 8 月 10 日发布版本 2.0.0。部署方式由 1.x 的 Helm chart 全面变更为 Operator。

## 致谢

首先感谢 @andyli029 @zhyass @runkecheng @acekingke @hustjieke

@molliezhang 提交的修改。

以下是完整的 Release Notes。

## 2.0.0 Release Notes

### Improvements

- Add post-start and pre-stop script [#155](#)
- Add PreStop for xenon container [#145](#)
- Move the charts images and change the key word [#140#142](#)
- Support roll update [#133#121](#)
- Unit test for container, cluster [#131#130](#)
- Add the document about the deployment of operator version [#132#127](#)
- Update the path of helm chart [#126#129](#)
- Update mysql version to 5.7.34 [#124#123](#)
- Add status api to support update the cluster status [#120#119](#)
- Add operator sidecar [#120#117](#)
- Update the config files, helm files, the Dockerfile, Makefile [#120](#)
- Update kubebuilder from v2 to v3 [#114#113](#)
- Modify the repo [#112](#)
- Adjust the dir for operator [#111](#)
- Add operator init [#123#109](#)
- Add rolling update feature code annotation [#165](#)
- Add ignore dir vendor and testbin [#153#154](#)



## Bug Fixes

- Fix the auditLog container [#181#179](#)
- Fix the incorrect description about MetricsOpts [#177](#)
- Fix the bug about PostStartHookError that command `sh -c /scripts/post-start.sh` exited with 126 [#171](#)
- Fix the path from docker to radondb [#167](#)
- Fix the bug about the pods 's status when the yaml have been changed [#166#164#161#158](#)
- Fix the bug that xenoncli cannot create user [#163#162](#)
- Fix the bug about reflect.SliceHeader vet error when go 1.16.6 [#141#139](#)
- Move the init.sql to mysql config dir radondb [#128](#)
- Fix the bug that innodb\_buffer\_pool\_size cannot be set correctly when its size greater than int32 [#125](#)

## 1.0

RadonDB MySQL on Kubernetes 1.0.0/1.1.0/1.2.0 是早期由 Helm Chart 方式部署的 3 个版本。

**强烈建议试用 2.0 最新版本。**

## 致谢

首先感谢 @andyli029 @zhyass @hustjieke @runkecheng @molliezhang @KID-G 提交的修改。

以下是完整的 Release Notes。

## 1.2.0 Release Notes

### Improvements

- Move dockerfile to dockerfiles [#108](#)
- Update logo\_randondb.png and modify files [#110](#)
- Add wechart community pic [#107]()
- Remove the step to configure-docs for the root password [#105](#)
- Update the architecture figure [#102](#)

### Bug Fixes

- Modify deploy links [#99](#)
- Fix some errors adjust some descriptions in README [#96](#)

## 1.1.0 Release Notes

### Improvements

- Add table content for each files [#98](#)
- Add deploy links on README.md and README\_zh.md [#97](#)

- Split the deploy-document according to the different deployment methods #95 #94
- TEST Issue template #92
- Add pull request and issue templates #91 #90
- Add the document to deploy radondb-mysql #89 #49 #45
- Add the network configuration document of the service #85
- Support the feature for k8e app #83
- Rename xenondb to randondb-mysql #77 #75 #74
- Modify the key word #73 #47 #41
- Add the README.md and README\_zh.md #63 #57 #55 #50 #48 #42 #37
- Support the feature for k8s #62
- Rename krypton to xenondb #40 #36
- Add publishNotReadyAddresses param in headless service #34
- Add CMD about Kubernetes #29 #21 #20 #17
- Add directory about test #16
- Support view mysql slow log #14
- Support 1 replica #13 #11
- Support read/write splitting #9
- Add the Steps about setup service for client to write/read #8
- Add remove lost+found in charts file #5
- Update the NOTES.txt #64 #3
- Add charts and dockerfile #34 #23 #18 #15 #1

## Bug Fixes

- Fix the error file name #93
- Modify the description in charts file #81 #66 #67 #68
- Modify the community info in READMME.md #78 #70 #69 #61 #60 #59 #52 #51
- Fix xenon error log #33 #32
- Fix the jump #31
- Fix the bug about sysbench FATAL: mysql\_stmt\_prepare() failed #25
- Fix the bug about hang when run cmd kubectl delete pv #24
- Fix the error about lint #22
- Fix the bug that execute sql with no response #18
- Fix the bug that slave-pod failed to initialize relay log info structure from the repository #12 #10
- Fix the path bug #7
- Fix the bug that install helm failed #4

## 1.1.0 Release Notes

### Improvements

- Move dockerfile to dockerfiles [#108](#)
- Update logo\_randondb.png and modify files [#110](#)
- Add wechart community pic [#107]()

- Remove the step to configure-docs for the root password [#105](#)
- Update the architecture figure [#102](#)

## Bug Fixes

- Modify deploy links [#99](#)
- Fix some errors adjust some descriptions in README [#96](#)

## 1.0.0 Release Notes

XenonDB is a Highly available cluster solutions that is based on MySQL database.

- Non-centralized automatic leader election.
- Second level switch
- Strongly consistent data
- Cluster management
- Logs, Monitoring and alerting
- Account management

XenonDB 为早期项目名，后更名为 RadonDB MySQL Kubernetes。