Kubernetes集群资源配置数据的备份、恢复和自动化

# 1、概述

1.1 Kubernetes是什么？

“Kubernetes（常简称为K8s）是用于自动部署、扩展和管理容器化（containerized）应用程序的开源系统。该系统由Google设计并捐赠给Cloud Native Computing Foundation（今属Linux基金会）来使用。”这是维基百科上对Kubernetes的介绍。

“它将组成应用程序的容器组合成逻辑单元，以便于管理和服务发现，Kubernetes 构建在 Google 15 年生产环境经验基础之上,并结合来自社区的最佳创意和实践。”这是Kubernetes官网中文版写在扉页上的寄语。

简单来说，Kubernetes能帮你把应用封装在容器里，想让它运行多少个副本就运行多少个。你的应用也许服务于1万个客户，部署10个应用容器，就能满足需求。也许你的应用服务于1000万客户，可能部署1000个应用容器就能满足需求。Kubernetes能帮你实现业务规模的自动水平伸缩。当然，Kubernetes能做的远不止这些。

Kubernetes对云计算的重要性，不亚于二十年前Java语言随第一波互联网浪潮掀起的革命性风暴。Java语言能让程序员的源代码“编写一次，运行在任何地方。”而Kubernetes引领的云计算革命，能让应用程序经过编排、调度组合，组成收放自如的复杂计算机应用系统。全自动绣花机把一根根线头、一个个色块、一个个印象元素，编织出绚丽的彩色图案。

某年双十一前，某宝的高管豪言，系统已经就绪，只需一杯清茶，再大的访问流量（N亿级用户），运维团队也能应对。难道他们没有用到可伸缩的容器编排？

1.2 为什么备份？

备份的目的是防止数据永久丢失，快速恢复系统可用。数据为什么会丢失呢？用户手潮误操作、不完美的版本升级、单点失效的硬件故障，地震、台风、洪水、海啸等自然灾害，都可能影响系统正常运行，造成数据部分丢失甚至全部灭失。

Kubernetes也是软件，软件运行的中间状态和运行结果也是以数据保存的。Kubernetes的运行数据存储在etcd，etcd是以键值对Key/Value格式存储，并提供读和写服务。

Etcd一般以多点集群形式出现，发生单点失效的可能性很小，但是用户误操作、升级失败的可能性还是存在的。

1.3 备份方法介绍

数据备份大致可以分为两类，物理备份和逻辑备份。

物理备份不区分数据的内在逻辑关系，把数据存储作为一个整体来备份，恢复时也是作为整体恢复，不能只恢复一部分数据。逻辑备份按照数据的内在逻辑关系，选择性提取部分数据或全部数据，恢复时可以选择恢复一部分数据。

数据的物理备份，一般来说对人类是不可读的，只有原软件系统才能读取识别。Oracle、MySQL的Dump文件、重做日志和数据快照，etcd、Redis数据文件的直接拷贝或者数据快照，都可以看做是物理备份。

从物理备份恢复数据的优点是明显的，要恢复完整数据很快。缺点也很明显，可能会丢失上次备份以来的数据更新，但是可以通过全量备份配合增量备份来弥补，缩小数据丢失的范围。物理备份也不能做部分恢复，那怕只是一小点数据更新。

从逻辑备份恢复数据的优点也是明显的，恢复部分数据很快，想要恢复哪部分就恢复哪部分。缺点是，逻辑备份的速度慢一些，遇到数据量大时耗时相对较长。

物理备份与逻辑备份的优缺点正好相反，在实践中不妨结合起来使用，扬长补短，发挥各自的优势。物理备份适合系统崩溃后的快速恢复重建，而逻辑备份适合更加精细化的局部修补。

本小文要介绍的Kubernetes数据备份是一种逻辑备份方法，需要深入到Kubernetes内部的逻辑结构，探寻奥秘，条分缕析，庖丁解牛，找到一种逻辑备份新途径。

# 2、模型

2.1 Kubernetes集群

2.2 Kubernetes层级模型

Kubernetes能支撑庞大而又复杂的应用系统，许多用户和团队共享集群，难免会相互影响或冲突。为了避免用户之间的冲突，Kubernetes引入命名空间概念， 在同一个命名空间下各种资源的不能重名，在不同的命名空间下允许重名。用户在分配给他的命名空间下操作，不用担心影响到别人，也不用担心受别人影响，因为每个用户或者每个团队都有独立的命名空间。

Kubernetes拥有和保留系统级的命名空间Kube-system和kube-public，未经授权不允许普通用户使用系统命名空间。默认命名空间default是公共的，如果没有指定命名空间，用户新建的资源都将建立在default命名空间。

Kubernetes的资源类型包括：服务 service、部署deploy、配置configmap、加密配置secret、任务job、定时任务cronjob、副本集replicaset、驻留任务集daemonset、有状态集statefulset等等，此处不一一列举。

Kubernetes的资源配置告诉Kubernetes系统，部署（deploy）哪些应用，对外提供哪些服务（service），应用运行参数（configmap）存在哪儿，敏感参数（secret）需要加密吗，运行一次就结束的任务（job）如何调度，像闹钟一样的定时任务（cronjob）怎么安排，一个节点运行一个且只运行一个的驻留任务（daemonset）支持吗，运行过后希望保留数据及状态的任务（statefulset）。资源配置赋予Kubernetes丰富的资源创建能力，能适应复杂多变的应用环境。

同一个资源类型下，可以配置多个不同的资源实例，而同一个资源实例允许运行多个完全相同的副本，拓展了系统服务能力。例如：供客户使用的nginx/http服务和供内部运营人员使用nginx/http服务，可以分别配置、分别部署，多副本运行。

如上所述，Kubernetes系统可以归纳为多层的分层模型，从上到下分别是Kubernete平台、命名空间namespace、资源类型resource type和资源实例resource instance。上一级和下一级之间是一对多的关系，而下一级从属于唯一的上级。例如，资源类型为deploy的应用实例nginx-web从属于类型deploy，而deploy从属于某个命名空间ns-cmft。命名空间可以容纳deploy、service和job等多种类型的资源。资源实例的副本，是运行时的概念，可以动态创建、动态销毁。

Kubernetes资源配置构成容器编排的身体骨架，而血肉在运行时填充塑形。或者说，资源配置就像是一个具体Kubernetes运行时实例的DNA、基因组。如果所有资源配置确定了，系统的内部结构也基本确定了。这个Kubernetes基因组的另一部分是容器镜像，以及容器运行时产生的数据。

Kubernetes资源结构的分层模型如下图所示：

命名空间：

几个系统命名空间

应用命名空间

资源从属于命名空间

资源类型：

服务 service

部署 deploy

配置 configmap

加密配置 secret

任务 job

定时任务 cronjob

副本集 replicaset

驻留任务集 daemonset

有状态集 statefulset

service deploy configmap secret job cronjob replicaset daemonset statefulset

2.3 数据备份模型

Kubernetes的资源配置可以从系统内导出来，存为yaml格式的文本文件，也可以根据导出来的yaml文件，重新建立与导出时完全一样的同名资源。这是Kubernetes备份和恢复的基本原理和技术基础，也是本文描述的核心过程。

前文说到Kubernetes资源是分层模型，从上到下是一对多的关系。那我们从上到下，按图索骥、顺藤摸瓜，一级一级往下探索，就能找到所有的命名空间、资源类型、资源实例，达成全面备份资源配置的目的。Kubectl是Kubernetes提供给管理用户使用的实用命令，它能查询命名空间、资源实例等，资源类型是固定的，作为查询参数传入。kubectl命令正好能满足我们的需求。

## 查询全部命名空间

# kubectl get ns

## 查询某类型下的资源列表

# kubectl -n namespace get deploy

## 查询具体资源的详细数据

# kubectl -n namespace get deploy nginx-web -o yaml > deploy\_nginx-web.yaml

## 从yaml文件重建资源实例

# kubectl -n namespace create -f deploy\_nginx-web.yaml

代码块中的namespace和nginx-web需要替换为具体命名空间和资源名称。

时间标尺快照

备份模型

2.4 数据恢复模型

数据恢复是从已备份的数据副本恢复到正在运行的Kubernetes系统。

这里的数据副本是指此前备份导出来的yaml格式文本文件。Yaml文件是人类肉眼可读的，也是可以修改的。如果管理员想恢复某个历史时点备份的资源文件，且修改某个错误值，那么从备份的yaml文件恢复是比较好的选择。

数据恢复可以是全量恢复，或者部分恢复。全量恢复是从某个时刻的全量备份数据恢复全部数据至Kubernetes系统。部分恢复只恢复部分数据。

相比数据备份的豪气、大方，数据恢复则要小心谨慎得多。数据恢复一般以部分恢复为主，只对发生故障，而且确认副本数据正确有效时才会恢复。全量恢复则是在系统发生不可逆转的全面崩溃时才会考虑，而且会优先考虑从物理备份恢复，在物理备份不可用时，才考虑逻辑备份（本文所说的yaml备份）。有时物理备份与逻辑备份配合使用恢复系统也是不错的选项。

部分恢复应当把经过仔细审查通过的yaml文件复制到专门的恢复目录（restore），以便于按顺序批量执行，恢复系统数据。

数据恢复应当记录详细的日志，以便事后查询、审计。

# 3、数据备份

3.1 全量备份

3.2 优先备份

3.3 备份流程

3.4 备份代码详解

3.5 备份的备份

# 4、数据恢复

4.1 准备数据

4.2 执行恢复

4.3 恢复检测

4.4 恢复代码详解

# 5、应用

5.1 故障恢复

5.2 历史跟踪

持续跟踪资源配置历史

5.3 时点比对

时点比对配置变化

# 6、自动化

6.1 备份自动化

数据备份的自动化

6.2 监测自动化

重点配置项监测的自动化。

# 7、多集群

7.1 多集群模型

7.2 多集群备份

7.3 多集群恢复

# 8、小结

8.1 工作小结

8.2 未来展望