大家好，我叫唐小吉，来自浙江杭州。

今天我要跟大家分享的内容是一些我个人在cockroachdb社区的工作经历以及用kubernetes搭配cockroachdb部署解决数据中心状态管理和数据高可靠性的实现。

。。。工作分享。。。

众所周知，cockroachdb是一个开源、分布式的、具有高可扩展性和极强生命力的数据库系统，但是，对于这种分布式的大型系统，部署和运维往往是一件比较棘手的事情。面对这种问题，我们自然而然就会想到当下最流行的容器解决方案以及我今天要讲的容器编排系统--kubernetes。提到容器、容器化，自然逃不开Docker，作为现在最流行的容器化解决方案，Docker是一个开源的容器引擎，可实现虚拟化，通过Docker，我们可以非常方便快速地搭建和移植跨平台应用。

cockroachdb的官方网站的文档中已有用docker安装cockroachdb详细的步骤，这里先不详细说，

但是，当我们进入cockroachdb官方文档中关于用docker安装cockroachdb的说明页面时，在页面顶部非常显眼的位置标有一个用红色标注的worning：

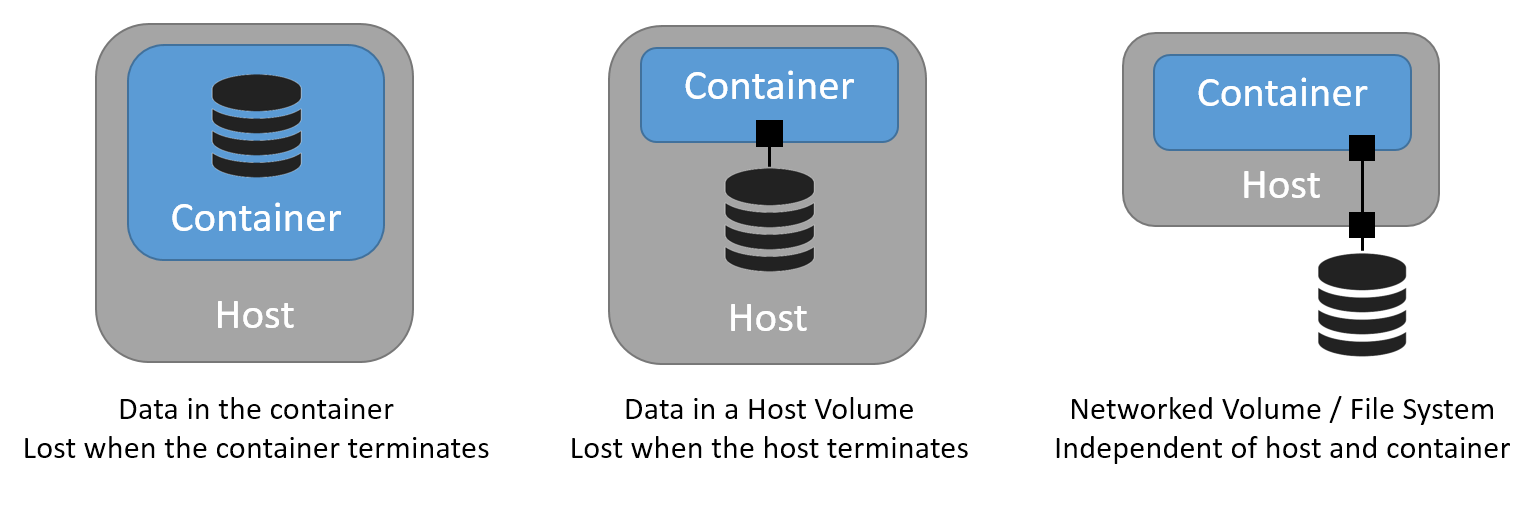


这里面竟然不建议用户用Docker的方式安装cockroachdb，那这又是为什么呢？

从这两句话里面可以看出来，官方不建议用Docker的方式安装cockroachdb的原因貌似是cockroachdb是\*\*有状态的\*\*应用，那么，什么是有状态的应用？或者说，什么是状态？

简单来说，状态就是应用程序运行所需的各种数据，包括但不限于程序的资源、配置、会话、连接、等，这么说来，应用本身其实都是有状态的，但是，在微服务时代，应用被设计为无状态的，因为要进行分布式的部署和运维，比如通过Docker进行容器化的部署，一般会把数据和应用进行隔离，但这并不是说把这些数据从应用中剥离，而只是将这些数据通过独立于应用的外部存储比如数据库、缓存、文件或其他存储的形式进行转存，哪怕程序要用到这些数据，也只是暂时将数据加载，也就是说应用只负责状态的搬运和改变，最终被改变的数据依然要被写到原来的存储单元中。这好比一个人只干活而没有记忆，当需要记录一些事情的时候，需要将相关内容写在纸上。

再返回来看用Docker部署cockroachdb这件事，这好像确实与无状态应用的理念背道而驰，因为作为一个数据库，cockroachdb主要的功能就是作为状态（数据）保存的单元，而此时要将状态独立于这个体系确实会是一个棘手的问题。

我们来设想几个场景： 

假设一个数据库进行无状态部署，我们可能需要考虑一些问题，比如数据存在哪里？容器的生命状态会更跟随宿主机，这里Docker提供的存储卷映射机制可以将数据映射至本地文件系统，但如果数据库进行分布式部署，那每个节点的数据如要进行全量同步是否是实际的？如果要将数据存储至独立于主机的集群外部的存储单元又会有什么问题？数据的可靠性、安全性又该如何保证？等等等等......

这么看来，cockroachdb官方并没有开玩笑。

对于这些问题，目前，kubernetes已经给出了解决方案，其中主要的机制是StatefulSet和PV（Persistent Volumes）

一般来说，当使用kubernetes部署服务时，服务都是以一个一个的pod存在于kubernetes集群中，当发生错误时，有某个pod被系统回收，此时，按照部署时配置的pod数量，kubernetes会自动生成新的pod来顶替原来的pod，其中kubernetes提供了一个稳定的标识符机制，也就是说虽然pod在改变，但这个pod的唯一标识符是不变的，对使用者来说并没有两样，这对于数据库的一致性和分布式事务具有非常重要的意义。

kubernetes提供的另一个机制----Persistent Volumes（持久卷）解决了数据存在哪才靠谱的问题。这里给出的持久卷，也就是可以挂载在Kubernetes任意节点上的远程磁盘，Kubernetes会持续监听各个节点上的副本数量，当某个节点上副本没有达到要求的数量时，Kubernetes会为其自动补齐。这种策略也使得cockroachdb的数据具有很好的可移植性。但也正因为这些存储单元是远程的，这与使用本地存储相比成本会更高。

下面简单讲述一下讲述怎样通过Docker和Kubernetes，部署和管理CockroachDB：

首先就是安装Docker和Kubernetes，官方文档对此安装步骤都有详细的描述，步骤也很简单，这里不再赘述

我们使用cockroachdb官方给出的ymal配置文件对kubernetes和docker进行配置：https://github.com/cockroachdb/cockroach/blob/master/cloud/kubernetes/cockroachdb-statefulset.yaml ，下载配置文件，运行：

$ kubectl create -f cockroachdb-statefulset.yaml

稍后就可以看到cockroachdb的配置成功，

service "cockroachdb-public" created

service "cockroachdb" created

poddisruptionbudget "cockroachdb-budget" created

statefulset "cockroachdb" created

可以通过以下命令查看服务：

$ kubectl get services

NAME READY STATUS RESTARTS AGE

cockroachdb-0 1/1 Running 0 29s

cockroachdb-1 0/1 Running 0 9s

通过以下命令查看pod及其状态：

$ kubectl get pods

NAME READY STATUS RESTARTS AGE

cockroachdb-0 1/1 Running 0 1m

cockroachdb-1 1/1 Running 0 41s

cockroachdb-2 1/1 Running 0 21s

接下来可以尝试使用这个集群

$ kubectl run cockroachdb -it --image=cockroachdb/cockroach --rm --restart=Never -- sql --insecure --host=cockroachdb-public

等到集群提示创建完成，我们就可以像使用普通数据库那样使用cockroachdb

root@cockroachdb-public:26257> CREATE DATABASE bank;

CREATE DATABASE

root@cockroachdb-public:26257> CREATE TABLE bank.accounts (id INT PRIMARY KEY, balance DECIMAL);

CREATE TABLE

root@cockroachdb-public:26257> INSERT INTO bank.accounts VALUES (1234, 10000.50);

INSERT 1

root@cockroachdb-public:26257> SELECT \* FROM bank.accounts;

+------+-------------+

| id | balance |

+------+-------------+

| 1234 | 10000.5 |

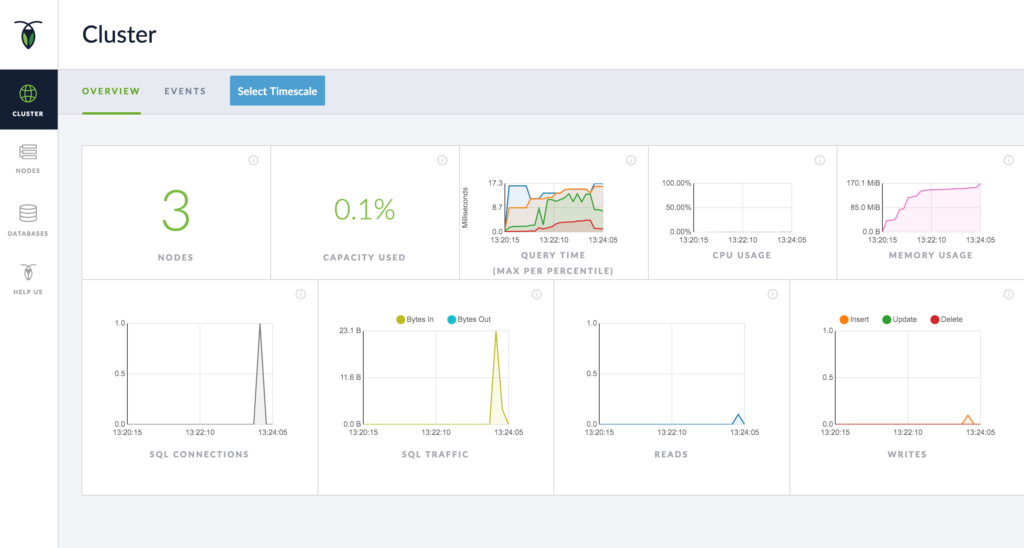
+------+----------+

(1 row)

另外，cockroachdb自带管理界面，运行：

$ kubectl port-forward cockroachdb-0 8080

就可以通过`http://localhost:8080/`来访问管理界面：

![admin-ui](/Users/txiaozhe/Documents/cockroach+kubernetes/admin-ui.png)

回顾我们上面提到的应用的状态管理，在kubernetes中已经将StatefulSet等组件集成到集群中了，针对我们上面提到的几个关于有状态的应用的问题，我们可以看一看效果如何：

可以模拟一次节点故障，比如运行：

$ kubectl delete pod cockroachdb-3

或

$ kubectl delete pod –selector app=cockroachdb

将其中一个pod或所有pod杀掉，但是过一会儿就可以看到新的pod被重启，因为这些被杀掉的pod通过StatefulSet 管理器被重新创建了，且其中的数据不会损坏，因为各自的数据已通过各自的Persistent Volumes（持久卷）恢复。

另外，kubernetes对集群的管理功能也是非常强大的，比如可以轻松的对集群进行缩放：

$ kubectl scale statefulset cockroachdb --replicas=4

或者关闭集群时按需删除创建的资源：

$ kubectl delete statefulsets,pods,persistentvolumes,persistentvolumeclaims,services,poddisruptionbudget -l app=cockroachdb

当然也可以一次性删除所有资源并关闭集群

$ gcloud container clusters delete cockroachdb-cluster

至此，通过kubernetes方式部署cockroachdb的流程就结束了，尽管目前来说cockroachdb的产品化还没有非常完善，但是我们仍然可以在cockroachdb上做很多事，不管是尝鲜还是探索，相信都会有所收获，也希望这样能让cockroachdb以及cockroachdb社区越来越好，我的分享结束了，谢谢大家！