LightDB K8S Guide

发布 *23.1*

LightDB

2023 年 05 月 20 日

目录:

1 前言

本文提供 LightDB 在 docker 及 K8S 中的部署方法。

2 K8S 安装包

LightDB 的 K8S 镜像包目前没有提供镜像仓库，需要下载镜像包后导入到您的环境中。

在安装之前需要获取对应版本的安装包，文件如下:

lightdb-operator.tar docker 镜像包

lightdb-patroni.tar docker 镜像包

config K8S 配置文件

3 镜像部署

LightDB 的 K8S 需要使用 `lightdb-patroni` 和 `lightdb-operator` 两个镜像, 例如:

```
lightdb-patroni:23.1
lightdb-operator:23.1
```

查看 K8S 集群节点列表

```
[root@master1 ~]# kubectl get node -o wide
NAME      STATUS    ROLES          AGE    VERSION    INTERNAL-IP     ...
master1   Ready     etcd,master    127d   v1.18.12   10.20.25.158    ...
node1     Ready     worker         127d   v1.18.12   10.20.25.162    ...
node2     Ready     worker         127d   v1.18.12   10.20.25.163    ...
node3     Ready     worker         127d   v1.18.12   10.20.25.173    ...
node4     Ready     worker         127d   v1.18.12   10.20.25.174    ...
node5     Ready     worker         127d   v1.18.12   10.20.25.176    ...
```

LightDB 使用离线的方式提供镜像包，需要导入到您的环境中，根据实际情况不同，可以使用 ctr,docker 或者镜像仓库的方式，确保 K8S 的每个节点能拉取到镜像就可以了。

3.1 使用 ctr 导入镜像

需要在 K8S 集群的每个节点都要导入镜像

使用 ctr 工具导入镜像，需要指定 namespace `k8s.io` , 以便 K8S 可以使用

```
ctr -n k8s.io images import  lightdb-operator.23.1.tar
ctr -n k8s.io images import  lightdb-patroni.23.1.tar
```

导入后，可以通过如下命令查看镜像是否已经存在

```
crictl images
```

3.2 使用私有镜像仓库部署

如果您有私有镜像仓库服务器，可将镜像推入到您的私有镜像仓库中，这可以避免在每个节点导入镜像的麻烦。

导入镜像仓库后，应将 `postgres-operator.yaml` 和 `minimal-lightdb-manifest.yaml` 两个配置文件中的镜像地址改为您的镜像仓库地址。例如: `image: 10.20.30.218:4983/lightdb-operator:23.1`，可以先手工使用 `crictl pull` 或者 `docker pull` 拉取镜像测试一下。

使用私有镜像仓库后，在集群启动时会自动拉取镜像，不再需要手工在每个节点单独部署镜像。

3.3 使用 docker 导入镜像

需要在 K8S 集群的每个节点都要导入镜像

```
docker load -i lightdb-patroni.23.1.tar
docker load -i lightdb-operator.23.1.tar
```

导入成功后，可以通过在每个节点执行下面命令确认镜像是否已经存在。

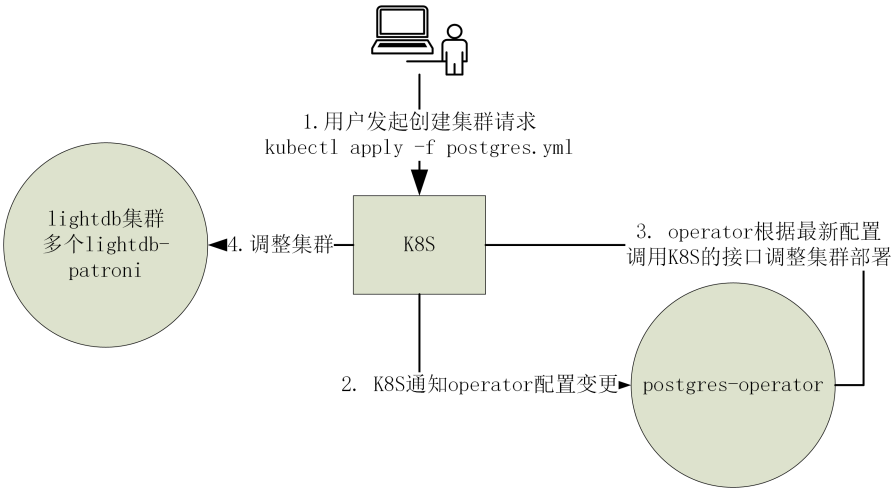
```
docker images | grep lightdb
```

注意:

1. 本版本仅支持在 x86 环境部署，不支持 arm，龙芯等环境。
2. 不建议使用 minikube 环境。

4 operator 部署

operator 负责根据用户的 yaml 配置修改 K8S 中的集群部署，例如用户修改或者创建 LightDB 集群的流程如下所示:



在部署好镜像的前提下，只需要应用一系列的 yaml 配置文件就可以完成 operator 的安装，具体步骤如下:

```
# registers the CRD
kubectl create -f config/operatorconfiguration.crd.yaml
kubectl create -f config/postgresql-operator-default-configuration.yaml
```

(下页继续)

(续上页)

```
kubectl create -f config/operator-service-account-rbac.yaml
kubectl create -f config/postgres-operator.yaml
```

在 postgres-operator.yaml 中有 operator 的镜像地址，见下方示例片段中的 image 字段，如果后续需要升级版本，则需修改此镜像配置。

```
# postgres-operator.yaml: 片段

containers:
- name: postgres-operator
   image: lightdb-operator:23.1
   imagePullPolicy: IfNotPresent
```

上面 yaml 应用完毕后，可以通过如下命令查看 deployment 和 pod 的情况确认 operator 是否正常运行。

```
# 查看deployment
$ kubectl get deployment
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
postgres-operator   1/1     1            1           33m

# 查看pod
$ kubectl get pod -l name=postgres-operator
NAME                                 READY   STATUS    RESTARTS   AGE
postgres-operator-5d8b76f5f6-svcjl   1/1     Running   0          110s
```

必须确认 operator 的状态是 Running 才可以进行下一步。

5 LightDB 集群部署

基于前面章节，operator 部署成功后，我们就可以部署 LightDB 集群了。

5.1 持久化存储配置

LightDB 的数据必须持久化存储。根据数据量和实例数量，创建一定数量的 PV，例如，数据库实例数据为 10GB，打算部署 1 主 1 备两个实例构成集群，则需要创建两个大小为 10GB 的 PV。

PV 的功能和相关的操作都是 K8S 直接原生支持，可以查看 K8S 官方文档存储相关章节了解更权威的信息: https://kubernetes.io/docs/concepts/storage/ 。

如何部署持久化存储不在本文档范围内，可参考相关手册配置合适的存储。下面是基于 NFS 共享存储的部署方法作为示例

```
apiVersion: v1
kind: PersistentVolume
metadata:
name: nfs-pv1
labels:
   pv: nfs-pv1
spec:
capacity:
   storage: 10Gi
accessModes:
   - ReadWriteOnce
```

(下页继续)

(续上页)

```
persistentVolumeReclaimPolicy: Retain
storageClassName: nfs
nfs:
   path: /data/nfsdata/v1
   server: 10.20.25.158
```

在 **部署好 NFS 服务器**后，可以根据您的实际情况修改此配置，然后使用 `kubectl apply` 命令创建 PV。创建好 PV 后，可以通过 `kubectl get pv` 查看已经创建的 PV 列表。

```
# 把上面的内容保存为文件，执行如下命令，创建PV
kubectl apply -f nfs-pv.yml
# 查看PV列表
kubectl get pv
```

如果您是其他形式的存储，则根据 K8S 官方文档指导创建 PV。

5.2 集群配置

LightDB 的集群配置文件为 `config/minimal-lightdb-manifest.yaml` 您需要根据实际部署情况修改配置项。

我们在这里先简要讲述一下配置文件中关键配置项的含义，后面有更详细的配置项说明。

一个最简单的 LightDB 配置文件如下所示:

```
apiVersion: "acid.zalan.do/v1"
kind: postgresql
metadata:
name: lightdbcluster77
spec:
teamId: "lightdb"
dockerImage: lightdb-patroni:22.4
volume:
   size: 1Gi
   # storageClass: nfs
numberOfInstances: 2
superUserPassword: 'abc123'
numberOfWorkNodes: 2
enableDistributed: true
postgresql:
   version: "13"
   parameters:
      ssl: "off"
patroni:
   initdb:
      install-mode: distributed
      compatible-type: oracle
      pwfile: "/home/lightdb/.pass"
   pg_hba:
   - local   all             all                                     trust
   - host    all             all                 127.0.0.1/32        trust
   - host    all             all                 0.0.0.0/0           md5
   - host    all             all                 ::1/128             md5
   - host   replication     standby               all                 trust
   - hostssl replication     standby              all                 trust
```

(下页继续)

(续上页)

```
  - host    all             all                 all                 md5
  - hostssl all             all                 all                 md5
```

需要关注的配置项含义如下:

1. **name** : 集群名称，如果需要创建多个集群，每个集群名称不能一样。
2. **dockerImage** : 对应 lightdb-patroni 的镜像, 需要根据部署时的 lightdb-patroni 版本修改, 可以通过 `docker image` 命令查看。
3. **volume** : 存储相关，其中 storageClass 需要和对应的 PV 对应，以便从指定的 PV 池中分配存储。如果在 minikube 测试环境中,storageClass 可以不配置，默认使用创建 emptydir 的临时存储。
4. **numberOfInstances** : 实例数量，一个主，其他是备；例如指定 2 表示一主一备，指定 3 表示一主二备。
5. **enableDistributed** : 是否部署分布式。
6. **numberOfWorkNodes** : 分布式 DN 节点数量, 仅在 enableDistributed=true 时生效，例如：配置为 2，表示是 1CN2DN 架构，其中 CN 和 DN 都是高可用架构 (1 主 1 备)
7. **patroni.initdb.compatible-type** : 兼容模式，可以：mysql,oracle,off 三种值，可根据实际情况选择。
8. **superUserPassword** 数据库 lightdb 用户密码，部署完成后
9. **storageClass** 和 pv 配置保持一致，可以从对应 pv 池中分配存储

其他未提及的配置项可以保持不变

5.3 创建集群

要创建 LightDB 集群，我们只需要创建一个配置文件，然后使用 `kubectl apply` 命令发给 K8S 即可。在安装包中已经包含了配置文件样例,

```
kubectl create -f config/minimal-lightdb-manifest.yaml
```

即可创建一个一主一备集群。

5.4 部署确认

1. 查看 lightdb 集群 CRD 状态, status 为 Running:

   ```
   [k8s@localhost operator]$ kubectl  get postgresql
   NAME               TEAM      VERSION   PODS   VOLUME  AGE     STATUS
   lightdbcluster77   lightdb   13        2      1Gi     3m20s   Running
   ```

   如果有异常，可以通过 `kubectl describe postgresql lightdbcluster77` 来查看错误详情。

2. 查看 POD 状态

   此时通过如下命令可以查看集群状态，如下所示，有三个高可用集群, 每个都是 1 主 1 从。

   lightdbcluster77-0 和 lightdbcluster77-1 是 CN, 其他两个集群是 DN 节点

   ```
   [root@master1 ~]# kubectl get pod -L spilo-role
   NAME                    READY   STATUS    RESTARTS   AGE   SPILO-ROLE


   lightdbcluster77-0      1/1     Running   0          38m   master
   ```

(下页继续)

(续上页)

```
lightdbcluster77-1      1/1     Running   0          38m   replica
lightdbcluster77-1-0    1/1     Running   0          35m   master
lightdbcluster77-1-1    1/1     Running   0          35m   replica
lightdbcluster77-2-0    1/1     Running   0          31m   master
lightdbcluster77-2-1    1/1     Running   0          31m   replica
```

注意 SPILO-ROLE 字段在容器创建过程中可能没有，此时表示高可用尚未创建完成，需要等 SPILO-ROLE 全部出来后才算正常。

如果有异常，可以通过如下命令查看错误信息:

```
# 查看POD节点日志
kubectl log lightdbcluster77-0

# 查看POD节点详情
kubectl describe pod lightdbcluster77-0
```

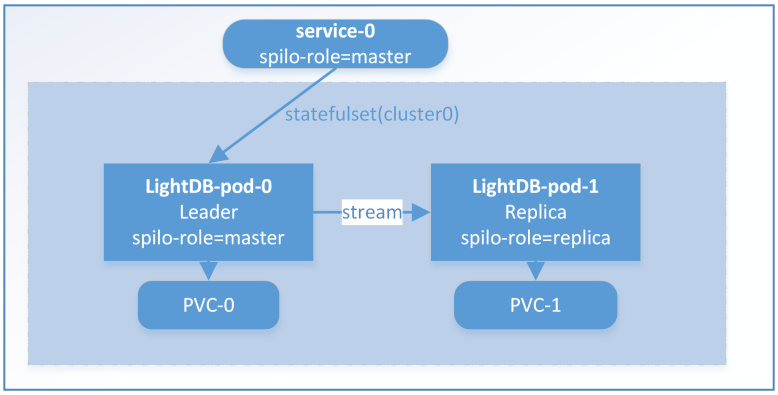
3. 进入节点查看数据库状态

```
# 进入节点容器
[k8s@localhost operator]$ kubectl exec -it lightdbcluster26-0 -- /bin/bash

# 在容器内查看
[root@lightdbcluster26-0 /]# patronictl.py -c /run/postgres.yml  list
+--------------------+------------+---------+---------+----+-----------+
| Member             | Host       | Role    | State   | TL | Lag in MB |
+ Cluster: lightdbcluster26 (7207906068718211118) ----+----+-----------+
| lightdbcluster26-0 | 172.17.0.6 | Leader  | running |  1 |           |
| lightdbcluster26-1 | 172.17.0.7 | Replica | running |  1 |         0 |
+--------------------+------------+---------+---------+----+-----------+
```

6 LightDB 集群组成

部署完成后，LightDB 在 K8S 中呈现如下架构 (以一主一从为例):



如上图所示，K8S 集群由 service, statefulset, pod,pvc 构成，并且依赖 K8S 内置的 ETCD(可选独立部署的 ETCD)。

6.1 postgresql

postgresql 代表 LightDB 集群。一个 LightDB 集群对应条记录。

```
[k8s@localhost operator]$ kubectl  get postgresql
NAME                TEAM      VERSION   PODS   VOLUME  AGE     STATUS
lightdbcluster26    lightdb   13        2      1Gi     3m20s   Running
```

如果想移除集群，使用如下命令删除集群即可:

```
kubectl  delete postgresql lightdbcluster26
```

6.2 service

service 是 K8S 内置的资源，用于向外提供 pod 的访问入口。因为 LightDB 的 pod 有主备之分，service 需要映射到正确的 pod 上，处理方法是通过 label 进行关联，当 lightdb-patroni 容器运行在主模式的时候，会设置自身标签 `spilo-role=master`；运行在备模式的时候，会设置自身标签为 `spilo-role=replica`，在定义 service 的时候，需要设置一个标签选择器，用于映射到正确的 pod 上。

可以通过如下命令查看 service:

```
[k8s@localhost operator]$ kubectl get service -l cluster-name=lightdbcluster26
NAME                        TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
lightdbcluster26            ClusterIP   10.103.88.200   <none>        5432/TCP   23m
lightdbcluster26-config     ClusterIP   None            <none>        <none>     23m
lightdbcluster26-repl       ClusterIP   10.96.231.217   <none>        5432/TCP   23m
```

上面代码可以看到一个 lightdb 集群创建了 3 个 service，其中 lightdbcluster26 指向主，lightdbcluster26-repl 指向备，lightdbcluster26-config 指向所有的 (本版本暂时没有用)。

默认创建的 service 类型是 ClusterIP, 仅在集群内部可访问。如果需要外部访问，可以考虑使用 Ingress 把服务暴露来。

测试环境暴露 Service

在测试环境下，可以使用如下方法暴露 service，

1. 另外创建一个 NodePort 类型的 servce 用于测试，参考配置文件如下:

```
apiVersion: v1
kind: Service
metadata:
   name: lightdb-test-service
spec:
   type: LoadBalancer
   selector:
      application: spilo
      cluster-name: lightdbcluster26
      spilo-role: master
   ports:
      - protocol: TCP
        port: 5432
        targetPort: 5432
        nodePort: 30001
```

2. 使用 kubectl 端口转发

把某个 pod 的 5432 端口映射到 6432, 外部可以访问对应主机的 6432 端口连到数据库。

```
kubectl port-forward --address 0.0.0.0  \
   pod/lightdbcluster26-0 6432:5432
```

6.3 statefulset

statefulset 是 k8s 自带的控制器，一个 LightDB 高可用集群的所有 pod 都在一个 statefulset 下。

```
[k8s@localhost operator]$ kubectl get statefulset
NAME               READY   AGE
lightdbcluster26   2/2     41m
```

6.4 POD

通过如下命令查看集群 pod，其中通过 spilo-role 字段可以看出主备，master 是主，replica 是备。

```
[k8s@localhost operator]$ kubectl get pod -L spilo-role
NAME                   READY   STATUS    RESTARTS   AGE     SPILO-ROLE
lightdbcluster26-0     1/1     Running   0          44m     master
lightdbcluster26-1     1/1     Running   0          44m     replica
```

6.5 PVC

LightDB 每个节点对应一个 PVC 请求，通过如下命令查看 PVC 状态，如果不是 Bound，则说明没有找到合适的存储，则需要确认 PV 的情况。

```
[k8s@localhost operator]$ kubectl get pvc
NAME                         STATUS   VOLUME                                     ...
pgdata-lightdbcluster26-0    Bound    pvc-65e897ff-f1ef-4116-b399-6b48f283622f   ...
pgdata-lightdbcluster26-1    Bound    pvc-46936dc9-3f93-4e06-beb4-236fa9daa5b3   ...
```

7 常见问题

7.1 POD 一直处于 Pending 状态

```
[root@node-1 wuxj]# kubectl get pod
NAME                READY   STATUS    RESTARTS  AGE
lightdbcluster13-0  0/1     Pending   0         94s
```

查看 pod 状态, 有提示 PersistentVolumeClaim 没绑定

```
[root@node-1 wuxj]# kubectl describe pod lightdbcluster13-0

<省略许多>

Events:
Type     Reason            Age   From               Message
```

(下页继续)

(续上页)

```
----     ------            ----  ----               -------
Warning  FailedScheduling  12s   default-scheduler  0/3 nodes are available: 3 pod␣
↪has unbound immediate PersistentVolumeClaims.
```

问题原因是：PVC 找不到对应的 PV，需要检查存储配置。

7.2 POD 状态为 ImagePullBackOff

```
[root@node-1 wuxj]# kubectl get pod
NAME                                    READY   STATUS             RESTARTS   AGE
postgres-operator-54ccd78fd5-r4rkf      0/1     ImagePullBackOff   0          57s
```

ImagePullBackOff 表示找不到镜像，可按如下步骤检查

1. 确认是否 K8S 集群的所有节点都导入了镜像，在 K8S 每个节点使用 `crictl images` 或者 `docker images` 确认镜像是否存在。
2. 确认 `postgres-operator.yaml` 和 `minimal-lightdb-manifest.yaml` 中的镜像配置是否和镜像仓库中的镜像名称一致。

   例如, 查看镜像信息如下

   ```
   [root@node-1 wuxj]# crictl images
   IMAGE                                TAG     IMAGE ID            SIZE
   docker.io/library/lightdb-operator   23.1    5088c15b8b98e       68.2MB
   ```

   此时应配置 `postgres-operator.yaml` 中的 `image` 字段为: `docker.io/library/lightdb-operator:23.1`