



链滴

Docker 系列之 MySQL 主从复制

作者: [hyboll](#)

原文链接: <https://ld246.com/article/1594608729565>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

 <https://ld246.com/images/img-loading.svg> alt="" data-src="https://b3logfile.com/bing/20180610.jpg?imageView2/1/w/960/h/540/interlace/1/q/100" data-bbox="145 65 285 90"/>

<blockquote>
<h2 id="目录">目录</h2>
</blockquote>

- 前言
- 主节点配置
- 从节点配置
- 扩展

<h2 id="前言">前言</h2>

<blockquote>
<h4 id="为什么基于Docker搭建-">为什么基于 Docker 搭建? </h4>
</blockquote>

- 资源有限，虚拟机搭建对宿主机配置有一定要求
- MySQL 安装步骤繁琐
- 一台宿主机可以运行多个 Docker 容器
- 容器之间相互隔离，有独立 ip，互不冲突
- Docker 使用简便，容器启动为秒级别

<blockquote>
<h4 id="MySQL-主从复制-也称-A-B-复制--的原理">MySQL 主从复制（也称 A/B 复制）的原理</h4>

</blockquote>
<p>MySQL 的主从复制方式有多种，本文主要演示基于日志 `binlog` 的主从复制式。</p>

- Master 将数据变更记录到二进制日志 `binary log` 中，也就是配置文件中 `log-bin` 指定的文件，这些记录叫做二进制日志事件 `binary log events`；
- Slave 通过 I/O 线程读取 Master 中的 `binary log events` 并写入到自己的中日志 `relay log` 中；
- Slave 重做中继日志中的事件，把中继日志中的事件信息一条一条的在本地执行一次，完成数据本地的存储，从而实现将改变反映到它自己的数据（数据重放）。

<blockquote>
<h4 id="启动容器">启动容器</h4>
</blockquote>

<p>首先启动两个 MySQL 容器，端口映射分别为 3307 (Master) 和 3308 (Slave)，具体的容器署请参考 <https://ld246.com/forward?goto=http%3A%2F%2Fblog.ovoll.cn%2Farticle%2F2020%2F07%2F03%2F1593757525484.html>。</p>

<p>PS: MySQL 容器的数据文件夹映射的路径不能相同</p>


<p> <https://ld246.com/images/img-loading.svg> alt="mysql.png" data-src="https://b3logfile.com/file/2020/07/mysql-8ba80b9c.png?imageView2/2/interlace/1/format/jpg" data-bbox="145 760 285 785"/>

<h2 id="主节点-Master-配置">主节点 (Master) 配置</h2>

<p>步骤简要说明</p>

- 配置数据库文件，指明 `server_id`，开启二进制日志 `log-bin`；
- 登录数据库，授予用户权限；

配置文件，而 `my.cnf` 配置文件中引入了 `conf.d` 文件目录，我们在创建容器的时候已经通过 `-v` 将该目录映射到了宿主机，所以我们只需要在宿主机的文件目录下添新的配置文件即可，无需进入容器中修改 `my.cnf` 文件。



2. 授权

`<blockquote>`

第一步，进入 MySQL 命令终端


`</blockquote>`

首先进入 MySQL 容器

```
docker exec -it mysql_3307 /bin/bash
```

然后进入容器后使用 root 用户登录 MySQL 进入命令终端

```
mysql -u root -p # 输入命令按回车键，然后输入root账号密码登录
```



`<blockquote>`

第二步，创建用于复制操作的用户并授权

`</blockquote>`

```
CREATE USER 'user'@'%' IDENTIFIED WITH mysql_native_password BY 'password'; # 由创建容器时已经创建过相同用户，此处不再创建
```

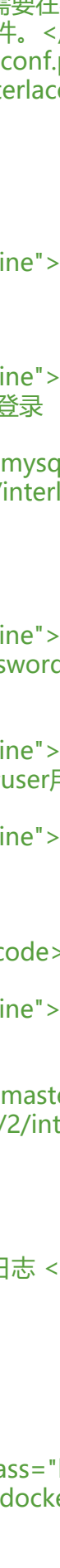
```
GRANT REPLICATION SLAVE ON *.* TO 'user'@'%'; # 授权给用户
```

```
flush privileges; # 刷新权限
```

3. 查看主节点信息

此时不要再对主节点做任何数据库操作，避免引起 `file` 和 `position` 变化

```
SHOW MASTER STATUS;
```



从节点 (Slave) 配置

步骤简要说明

``

配置数据库文件，指明 `server_id`，开启二进制日志 `log-bin`；

登录数据库，设置主节点 (Master)；

Slave 相关命令

``

1. 配置数据库文件

进入容器映射到宿主机中的 MySQL 配置文件目录

```
cd /usr/docker/mysql_3308/conf <
```

```
pan class="highlight-c1"># MySQL配置文件目录, 创建容器时, 通过'-v /usr/docker/mysql_330
/conf:/etc/mysql.conf.d'映射的文件目录</span>
</span></span></code></pre>
<p>创建 <code>my_conf.cnf</code> 文件</p>
<pre><code class="language-bash highlight-chroma"><span class="highlight-line"><span c
lass="highlight-cl">touch my_conf.cnf <span class="highlight-c1"># 创建自定义配置文件</s
an>
</span></span></code></pre>
<p>在 <code>my_conf.cnf</code> 文件中添加配置</p>
<pre><code class="language-bash highlight-chroma"><span class="highlight-line"><span c
lass="highlight-cl"><span class="highlight-o">[</span>mysql<span class="highlight-o">]<
span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
light-c1"># 同一局域网内注意要唯一</span>
</span></span><span class="highlight-line"><span class="highlight-cl">server-id<span cl
ss="highlight-o">=</span><span class="highlight-m">2</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
light-c1"># 开启二进制日志功能, 可以随便取 (关键) </span>
</span></span><span class="highlight-line"><span class="highlight-cl">log-bin<span clas
s="highlight-o">=</span>mysql-bin
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
light-c1"># 可选, 日志的过期时间</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
light-nv">expire_logs_days</span> <span class="highlight-o">=</span> <span class="highl
ght-m">10</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
light-c1"># 可选, 日志的最大大小</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
light-nv">max_binlog_size</span> <span class="highlight-o">=</span> 200M
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
light-c1"># 可选, 设置从节点 (Slave) 只读, 不可写</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
light-nv">read_only</span><span class="highlight-o">=</span><span class="highlight-m"
1</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
light-c1"># 可选, 同步的数据库名称, 可填多项</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
light-nv">replicate_do_db</span> <span class="highlight-o">=</span> test_1_db
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
light-c1">#replicate_do_db = test_2_db</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
light-c1"># 可选, 忽略同步的数据库, 可填多项</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
light-nv">replicate_ignore_db</span> <span class="highlight-o">=</span> ignore_1_db
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
light-c1">#replicate_ignore_db = ignore_2_db</span>
</span></span></code></pre>
<p>配置添加完成后, 重启 MySQL 容器</p>
<pre><code class="language-bash highlight-chroma"><span class="highlight-line"><span c
lass="highlight-cl">docker restart mysql_3308
</span></span></code></pre>
<p><strong>2. 配置主节点 (Master) </strong></p>
<p>进入从节点 (Slave) MySQL 命令终端, 然后设置主节点 (Master) 参数</p>
```



```
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight-cl">change master to master_host='172.17.0.3',
</span></span><span class="highlight-line"><span class="highlight-cl">  master_port=3
06,
</span></span><span class="highlight-line"><span class="highlight-cl">  master_user='u
er',
</span></span><span class="highlight-line"><span class="highlight-cl">  master_passwo
d='password',
</span></span><span class="highlight-line"><span class="highlight-cl">  master_log_file
'mysql-bin.000004',
</span></span><span class="highlight-line"><span class="highlight-cl">  master_log_po
=2774;
</span></span></code></pre>
```

<p>命令说明</p>

```
<pre><code class="language-properties highlight-chroma"><span class="highlight-line"><span class="highlight-cl">master_host      # Master的地址, 指的是容器的独立ip, 可以通过 'do
ker inspect --format='{{.NetworkSettings.IPAddress}}' 容器名称|容器id' 查询容器的ip
</span></span><span class="highlight-line"><span class="highlight-cl">master_port      #
Master的端口号, 指的是容器的端口号
</span></span><span class="highlight-line"><span class="highlight-cl">master_user      #
用于数据同步的用户
</span></span><span class="highlight-line"><span class="highlight-cl">master_password
  # 用于同步的用户的密码
</span></span><span class="highlight-line"><span class="highlight-cl">master_log_file
# 指定Slave从哪个日志文件开始复制数据, 即上文中提到的File字段的值
</span></span><span class="highlight-line"><span class="highlight-cl">master_log_pos
# 从日志文件中的哪个开始读, 即上文中提到的Position字段的值
</span></span><span class="highlight-line"><span class="highlight-cl">master_connect_r
try # 如果连接失败, 重试的时间间隔, 单位是秒, 默认是60秒
</span></span></code></pre>
```

<p>查看从节点 (Slave) 状态</p>

```
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight-cl">show slave status \G
</span></span></code></pre>
```

<p>正常情况下, <code>SlavIORunning</code> 和 <code>SlaveSQLRunning</code> 都是 o, 因为我们还没有开启主从复制过程。</p>

<p></p>

<p>开启主从同步</p>

```
<pre><code class="language-bash highlight-chroma"><span class="highlight-line"><span class="highlight-cl">start slave<span class="highlight-p">;</span>
</span></span></code></pre>
```

<p>再次查询主从同步状态 <code>show slave status \G</code>, <code>SlavIORunning</code> 和 <code>SlaveSQLRunning</code> 都是 Yes, 说明主从复制已经开启, 此时可以测试数据步是否成功。</p>

<p></p>

<p>PS: </p>

从节点 (Slave) 只会同步生效这一时刻后的内容, 之前的主节点 (Master) 数据需要在配置同之前备份到从库中。

主从数据库若没有保持一致的话, 后续的同步过程中从库会发生异常。

从库发生异常处理后, 需要更新从库中主库的 <code>master_log_pos</code> 的值

