ClickHouse集群搭建从0到1

http://www.jianshu.com/p/ae45e0aa2b52?from=timeline&isappinstalled=0

阅读此文，你将得到什么：
1、ClickHouse安装的2种方法，以及背后的坑
2、一步步帮你实现ClickHouse从单机到集群化，以及集群化的原理、配置文件等
3、集群化的2种方案，孰优孰劣

组件介绍

- ClickHouse安装完后，会有几个重要命令：

  1 clickhouse-server ClickHouse的Server端，也就是CK数据库的核心程序，相当于mysqld命令，提供数据库服务端
  2 clickhouse-client ClickHouse自带的client端，提供命令行的交互操作方式，来连接服务端，相当于mysql命令

Docker安装

- 官方默认只支持Ubuntu，并且提供了Docker镜像：
  - ClickHouse Server
  - ClickHouse Client

- Docker安装很方便，但是有几个问题：
  - 默认监听了IPv6，如果你的服务器恰好没有开启V6，会导致Docker启动失败的
    - 解决方案：使用Docker命令，cp出默认的配置文件，修改network监听后，拷回Docker，重新启动即可
  - 默认时区问题并不是东八区，如果没有修改，一些时间函数会差8个小时
  - clickhouse-client在Docker下，无法正常输入中文（调了LANG，无效，如果搞定记得告诉我）
  - 数据目录如果有要求，额外在Docker启动的时候，挂载一下
  - 配置文件不方便修改

- 建议调整：
  - 挂载本地服务器时区，或者直接修改Docker里的时区文件
  - 拷贝所有配置文件到服务器目录，启动时做映射，方便修改
  - 映射一个专用的数据目录
- 由于Docker方式我并不在线上采用，这里不再举例。建议仅仅作为笔记本上测试、了解用（不过前期，没有找到rpm包，Docker的确帮了我们很大的忙）。

RPM包安装

- 不出意外的话，99%的服务器都是CentOS系列
- 官方没有提供rpm包，但是Altinity公司提供了，关于这个公司的介绍，可以参照我其他文章

- 如果下载不了，看我的百度网盘，密码yv72（我这里好久没更新了，请注意看版本）
- CentOS推荐7.3以上，基本没有依赖包的问题
- rpm -ivh * 即可安装完成

配置文件解析

- ClickHouse有几核心的配置文件：

  1 config.xml 端口配置、本地机器名配置、内存设置等
  2 metrika.xml 集群配置、ZK配置、分片配置等
  3 users.xml 权限、配额设置

- 以上文件都可以在官方git下载到

rpm启动方式

- rpm安装后，会在服务器上生成如下几个文件：

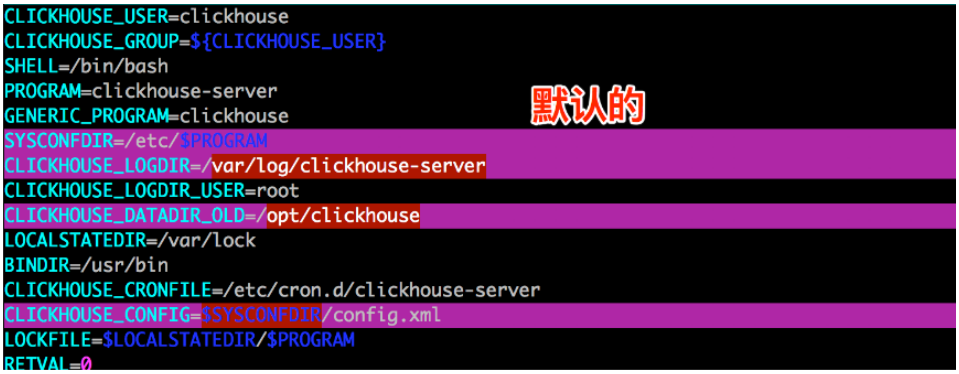
```
# 默认配置文件位置
root@localhost.localdomain:/  # ls /etc/clickhouse-server
config.xml  users.xml

# 上述文件定义了默认数据目录，临时目录位置，日志目录
/var/lib/clickhouse
/var/lib/clickhouse/tmp/
/var/log/clickhouse-server

# 默认启动脚本，注意，这个名字虽然叫server，其实是个shell脚本
/etc/rc.d/init.d/clickhouse-server
root@localhost.localdomain:/  # file /etc/rc.d/init.d/clickhouse-server
/etc/rc.d/init.d/clickhouse-server: POSIX shell script, ASCII text executable, with very long lines

# 最大文件打开数
root@localhost.localdomain:/  # cat /etc/security/limits.d/clickhouse.conf
clickhouse      soft    nofile  262144
clickhouse      hard    nofile  262144

# 默认crontab目录（没啥用）
/etc/cron.d/clickhouse-server

# 剩下就是/usr/bin下的二进制文件，但其实都是软链接到了clickhouse这个二进制文件
root@localhost.localdomain:/usr/bin  # ll | grep click -i
-rwxr-xr-x    1 root root      63M Sep 20 16:58 clickhouse
lrwxrwxrwx    1 root root       10 Dec 11 17:14 clickhouse-client -> clickhouse
-rwxr-xr-x    1 root root     3.3M Sep 20 16:58 clickhouse-compressor
lrwxrwxrwx    1 root root       10 Dec 11 17:14 clickhouse-server -> clickhouse
```

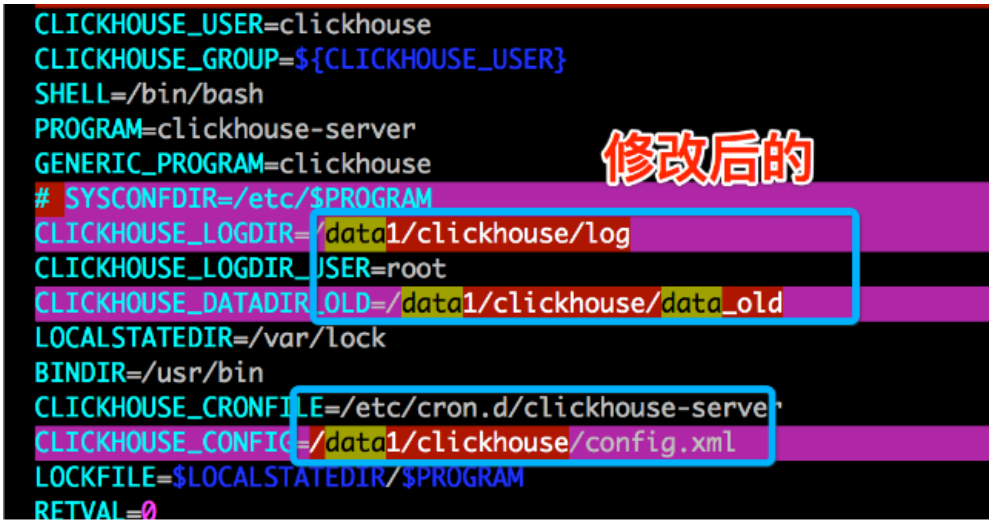
- 知道上述几个文件的作用后，我们就知道该怎么做了
  - 默认的数据目录明显不合理，特别是对于部分机器，系统盘和数据盘是不同的配置，需要单独挂载，以我们为例，我们统一使用/data1来放数据，数据目录以clickhouse命名，考虑到不用单机多实例，不以clickhouse${port}来命名

  - 默认的配置文件，对我们的管理也是个隐患，建议把配置文件、数据目录、临时目录、日志文件，统一放到/data1/clickhouse里，即：

```
root@localhost.localdomain:/data1/clickhouse  # tree . -L 1
├── config-preprocessed.xml
├── config.xml
├── cores
├── data
├── flags
├── log
├── metadata
├── metrika.xml
├── start_ck.sh
├── status
├── tmp
├── users-preprocessed.xml
└── users.xml
```

- 关于如何启动，我们的做法是：
    - 修改默认的shell脚本，修改默认配置文件的位置，即上面的start_ck.sh





- 这里其实是可以直接使用`clickhouse-server`（二进制那个），并采用`-d`参数启动的，但是实际过程，遇到了很多意外的情况，比如`-d`后，并不会以`daemon`方式启动，后来就不考虑直接命令行方式了
- 修改`config.xml`里对数据目录的定义

```
<?xml version="1.0"?>
<yandex>
  <!-- 日志 -->
  <logger>
     <level>trace</level>
     <log>/data1/clickhouse/log/server.log</log>
     <errorlog>/data1/clickhouse/log/error.log</errorlog>
     <size>1000M</size>
     <count>10</count>
  </logger>
```

```
    <!-- 端口 -->
    <http_port>8123</http_port>
    <tcp_port>9000</tcp_port>
    <interserver_http_port>9009</interserver_http_port>

    <!-- 本机域名 -->
    <interserver_http_host>这里需要用域名，如果后续用到复制的话</interserver_http_host>

    <!-- 监听IP -->
    <listen_host>0.0.0.0</listen_host>
    <!-- 最大连接数 -->
    <max_connections>64</max_connections>

    <!-- 没搞懂的参数 -->
    <keep_alive_timeout>3</keep_alive_timeout>

    <!-- 最大并发查询数 -->
    <max_concurrent_queries>16</max_concurrent_queries>

    <!-- 单位是B -->
    <uncompressed_cache_size>8589934592</uncompressed_cache_size>
    <mark_cache_size>10737418240</mark_cache_size>

    <!-- 存储路径 -->
    <path>/data1/clickhouse/</path>
    <tmp_path>/data1/clickhouse/tmp/</tmp_path>

    <!-- user配置 -->
    <users_config>users.xml</users_config>
    <default_profile>default</default_profile>

    <log_queries>1</log_queries>

    <default_database>default</default_database>

    <remote_servers incl="clickhouse_remote_servers" />
    <zookeeper incl="zookeeper-servers" optional="true" />
    <macros incl="macros" optional="true" />

    <!-- 没搞懂的参数 -->
    <builtin_dictionaries_reload_interval>3600</builtin_dictionaries_reload_interval>

    <!-- 控制大表的删除 -->
    <max_table_size_to_drop>0</max_table_size_to_drop>

    <include_from>/data1/clickhouse/metrika.xml</include_from>
</yandex>
```

单机

- 无需多解释，就是单机部署
- 按照上述方式安装rpm包，修改默认的config文件和启停控制脚本，启动即可
- 我上面的配置文件里，直接包含了集群的配置文件，如果只用了上述文件，是无法正常启动的
- 看这个文章的，应该都是冲着后面的集群搭建来的吧，所以，忽略这一个吧

分布式集群

CK是如何实现分布式的

- CK的分布式，完全依赖配置文件，即每个节点，都共享同样的配置文件，这个配置文件里，写了我跟谁是一个cluster的，我自己的名字是啥
- 如下面的配置文件里，有3个分片，各自用域名来标记，如果需要密码的话，集群也要写上明文密码和用户名
- 这样，就行程了ClickHouse的集群

- 集群怎么用?
    - 答案是指定引擎
    - CK里的引擎有十几个，这里只推荐3个：

    1 MergeTree，是CK里最Advanced的引擎，性能超高，单机写入可以达到50w峰值，查询性能非常快，有兴趣看我其他文章
    2 ReplicatedMergeTree，基于MergeTree，同时引入ZK，做了复制，下文会说
    3 Distributed，分布式引擎，本身不存储数据，可认为就是一张View，如果写入，会把请求丢到集群里的节点（有算法控制），如果查询，会帮你做查询转发再聚合返回

- metrika.xml

```
<yandex>
<!-- 集群配置 -->
<clickhouse_remote_servers>
    <bip_ck_cluster>
        <!-- 数据分片1  -->
        <shard>
            <internal_replication>false</internal_replication>
            <replica>
                <host>ck31.xxxx.com.cn</host>
                <port>9000</port>
                <user>default</user>
                <password>6lYaUiFi</password>
            </replica>
        </shard>

        <!-- 数据分片2  -->
        <shard>
            <internal_replication>false</internal_replication>
            <replica>
                <host>ck32.xxxx.sina.com.cn</host>
                <port>9000</port>
                <user>default</user>
                <password>6lYaUiFi</password>
            </replica>
        </shard>

        <!-- 数据分片3  -->
        <shard>
            <internal_replication>false</internal_replication>
            <replica>
                <host>ck33.xxxxa.com.cn</host>
                <port>9000</port>
                <user>default</user>
                <password>6lYaUiFi</password>
            </replica>
        </shard>
```

```
    </bip_ck_cluster>
</clickhouse_remote_servers>

<!-- 本节点副本名称（这里无用） -->
<macros>
    <replica>ck1</replica>
</macros>

<!-- 监听网络（貌似重复） -->
<networks>
   <ip>::/0</ip>
</networks>

<!-- ZK  -->
<zookeeper-servers>
  <node index="1">
    <host>1.xxxx.sina.com.cn</host>
    <port>2181</port>
  </node>
  <node index="1">
    <host>2.xxxx.sina.com.cn</host>
    <port>2181</port>
  </node>
  <node index="1">
    <host>3.xxxxp.sina.com.cn</host>
    <port>2181</port>
  </node>
</zookeeper-servers>

<!-- 数据压缩算法  -->
<clickhouse_compression>
<case>
  <min_part_size>10000000000</min_part_size>
  <min_part_size_ratio>0.01</min_part_size_ratio>
  <method>lz4</method>
</case>
</clickhouse_compression>

</yandex>
```

- user.xml
- 关于用户名密码的问题，在另一篇文章有解释，这里只贴上配置文件

```
<?xml version="1.0"?>
<yandex>
    <profiles>
        <!-- 读写用户设置  -->
        <default>
            <max_memory_usage>10000000000</max_memory_usage>
            <use_uncompressed_cache>0</use_uncompressed_cache>
            <load_balancing>random</load_balancing>
        </default>

        <!-- 只写用户设置  -->
        <readonly>
            <max_memory_usage>10000000000</max_memory_usage>
```

```
            <use_uncompressed_cache>0</use_uncompressed_cache>
            <load_balancing>random</load_balancing>
            <readonly>1</readonly>
        </readonly>
    </profiles>

    <!-- 配额 -->
    <quotas>
        <!-- Name of quota. -->
        <default>
            <interval>
                <duration>3600</duration>
                <queries>0</queries>
                <errors>0</errors>
                <result_rows>0</result_rows>
                <read_rows>0</read_rows>
                <execution_time>0</execution_time>
            </interval>
        </default>
    </quotas>

    <users>
        <!-- 读写用户 -->
        <default>
            <password_sha256_hex>967f3bf355dddfabfca1c9f5cab39352b2ec1cd0b05f9e1e6b8f629705fe7d6e</password_sha256_hex>
            <networks incl="networks" replace="replace">
                <ip>::/0</ip>
            </networks>
            <profile>default</profile>
            <quota>default</quota>
        </default>

        <!-- 只读用户 -->
        <ck>
            <password_sha256_hex>967f3bf355dddfabfca1c9f5cab39352b2ec1cd0b05f9e1e6b8f629705fe7d6e</password_sha256_hex>
            <networks incl="networks" replace="replace">
                <ip>::/0</ip>
            </networks>
            <profile>readonly</profile>
            <quota>default</quota>
        </ck>
    </users>
</yandex>
```

简单分布式方案

- MergeTree + Distributed

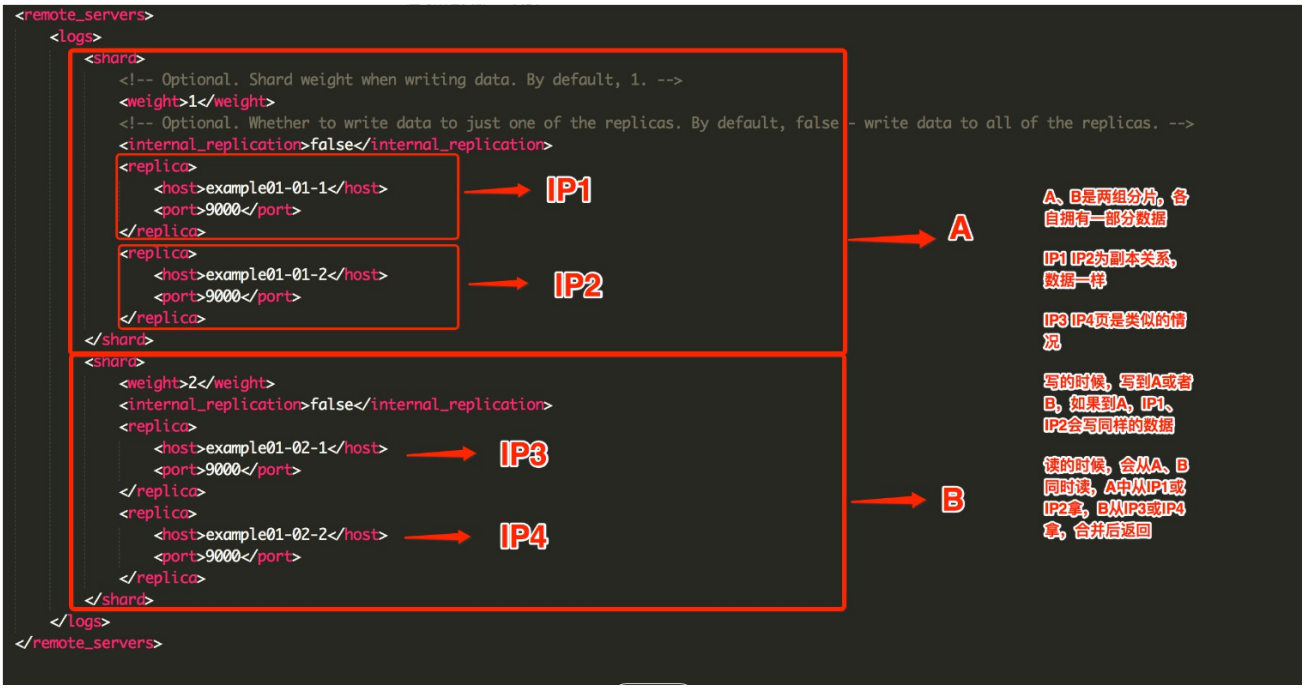
```
CREATE TABLE db.tb (date Date, ……) ENGINE = MergeTree(date, (date, hour, datetime), 8192)
CREATE TABLE db.tb_all (date Date, ……) ENGINE = Distributed(bip_ck_cluster, 'ck_test', 'dagger', rand())"
```

- db.tb为本地表，数据只是在本地
- db.tb_all为分布式表，查询这个表，引擎自动把整个集群数据计算后返回
- 像不像一台手动挡的车

分布式+高可用方案1

- 上述方案，使用过后，会发现CK的性能真的是超级快，这里我就不在贴图了，有兴趣可以看我那122页的PPT
- 但是有个问题，以上面3个节点为例，每个节点占1/3，如果宕机1个节点，直接丢掉1/3的数据，不能忍啊
- 于是，就得考虑数据的安全性，即副本
- MergeTree + Distributed + 集群复制

- 配置如下：



- 解释都在图里了，提一点，如果IP1挂了，IP2还在，不影响集群查询

- 这种方案为什么我们没有用呢?

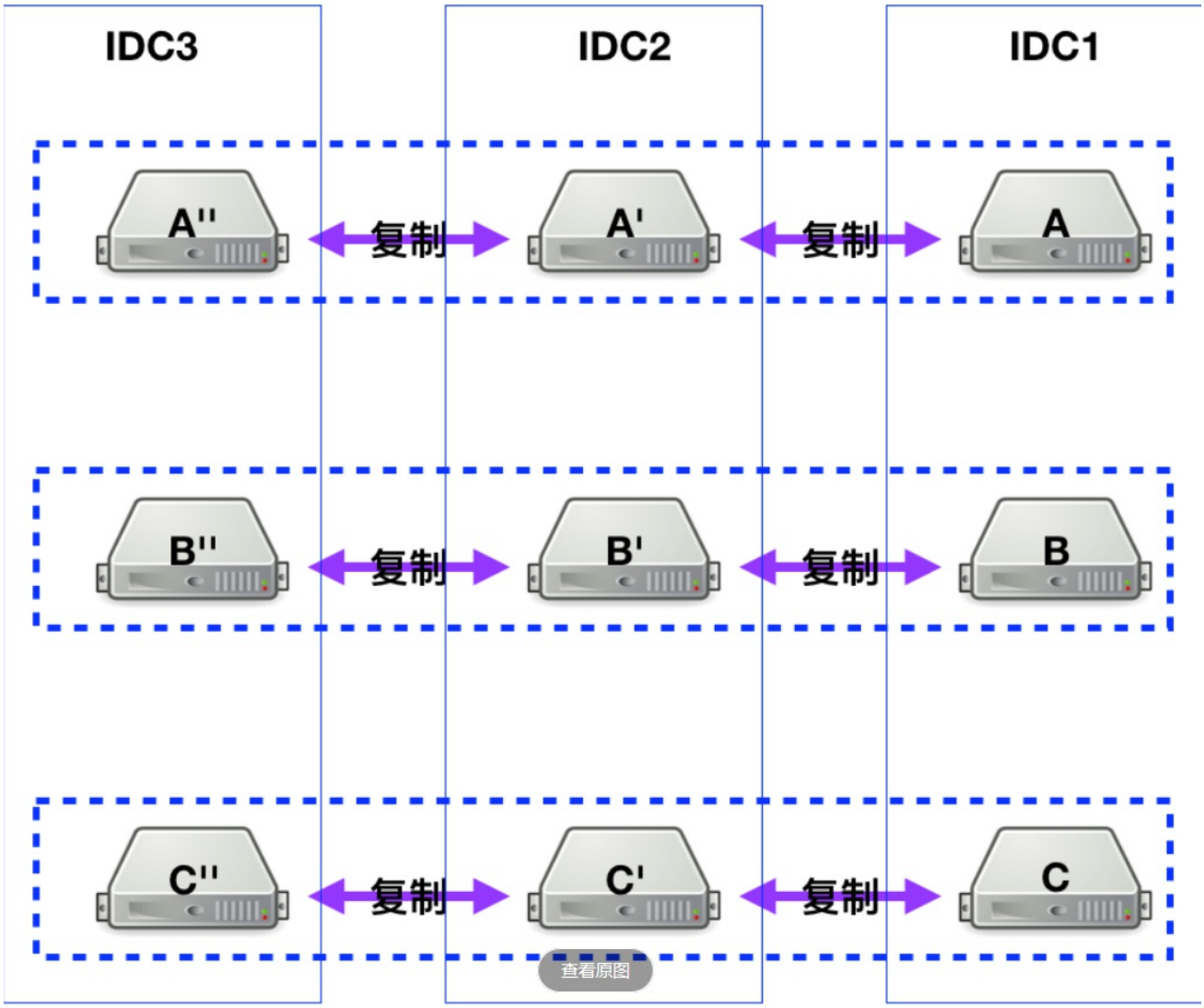
    - 如果IP1临时宕机，从宕机开始到恢复，期间的增量数据是可以补全的，依赖的IP2上的推送机制，会有临时目录
    - 但是，如果IP1彻底玩完，硬盘坏了，无法恢复，只能重做，引入一个IP5来替换IP1，这时候问题就来了，存量数据无法恢复
    - 这个方案之前有过争议，我坚持上面的观点，由于时间有限，没有详细测试，从CK原理上来讲，的确存在上述的问题，所以我们不用

分布式+高可用方案2

- ReplicatedMergeTree + Distributed
- 仅仅是把MergeTree引擎替换为ReplicatedMergeTree引擎
- ReplicatedMergeTree里，共享同一个ZK路径的表，会相互，注意是，相互同步数据

  CREATE TABLE db.tb (date Date, ……) ENGINE = ReplicatedMergeTree('/clickhouse/db/tb/name', 'node_name', date, (date, hour, datetime), 8192)
  CREATE TABLE db.tb_all (date Date, ……) ENGINE = Distributed(bip_ck_cluster, 'ck_test', 'dagger', rand())"

- 示意图架构就是这样：



- 每个IDC有3个分片，各自占1/3数据
- 每个节点，依赖ZK，各自有2个副本
- 这样，就不怕宕机啦~

CK分布式的问题

- 写哪个表
  - 可以写xxx_all，也可以写xxx本地表
  - 前者由于分布式表的逻辑简单，仅仅是转发请求，所以在转发安全性上，会有风险，并且rand的方式，可能会造成不均衡
  - 我们建议，通过DNS轮训，写本地表，这样最保险和均衡
- 读哪个表
  - 毫无疑问，是xxx_all表
- 3个节点，要么都用，要么都不用，不能只用2个或者1个
- 集群配置里，我们用了域名，本想着方便切换，但是CK只有在启动的时候，才会做解析
  - 那故障了怎么切换?
  - CK有一个厉害的地方，节点变动，无需重启，会自动加载
  - 利用上述特性，我们先去掉一个节点的配置，再加上这个节点的配置（DNS变更后），即可不重启就完成fail over

总结

- ClickHouse的性能令人印象深刻，但是，整个操作，又非常像一台手动挡的车，如果不是老司机，用着用着可能数据都没了，所以，掌握好原理，是开好这辆“超跑”的关键

- 上述集群中，你是否觉得表管理非常麻烦呢？的确，又要区分集群，又要区分副本，建议写一个脚本来统一建表，我们就是这么搞的