

基于 Go 的 MongoDB 实时同步工具 及 Docker 化实践

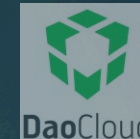


2017-02-25

马艳云

上海讯联数据服务有限公司

Magisync是什么



IT大咖说
不止于技术

Magisync 是用 Go 语言开发的 MongoDB 到MongoDB 或其他多种数据库之间的实时同步工具。



ORACLE®



为什么选择 Go

- ❖ 我们的核心交易系统就是 Go 语言开发的
- ❖ Go 语言相对来说语法比较简单，使用方便
- ❖ 在 GitHub 上有一个用 Go 语言开发的同步的程序



为什么开发 Magisync

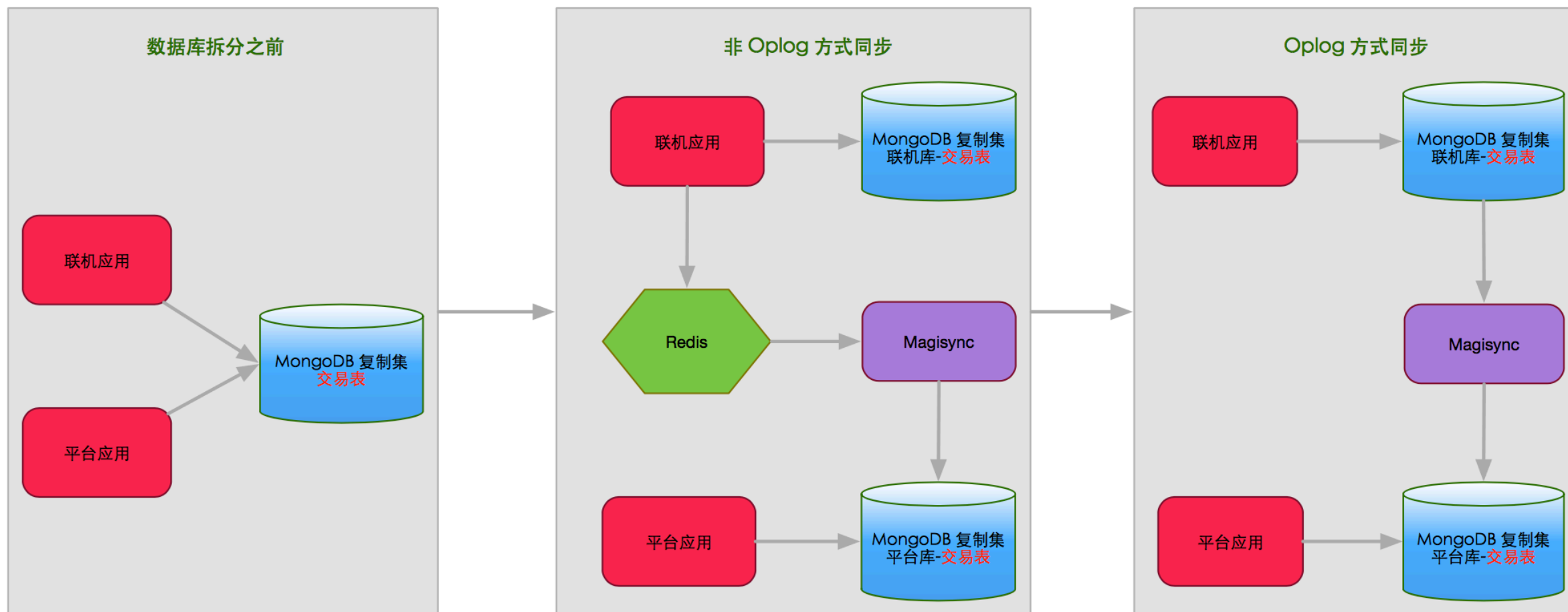


- ❖ 交易量日益增多，原有的系统架构不能满足业务需求
- ❖ 通过数据库拆分实现冷热数据分离
- ❖ 跨机房同步
- ❖ 市面上没有 MongoDB 同步工具

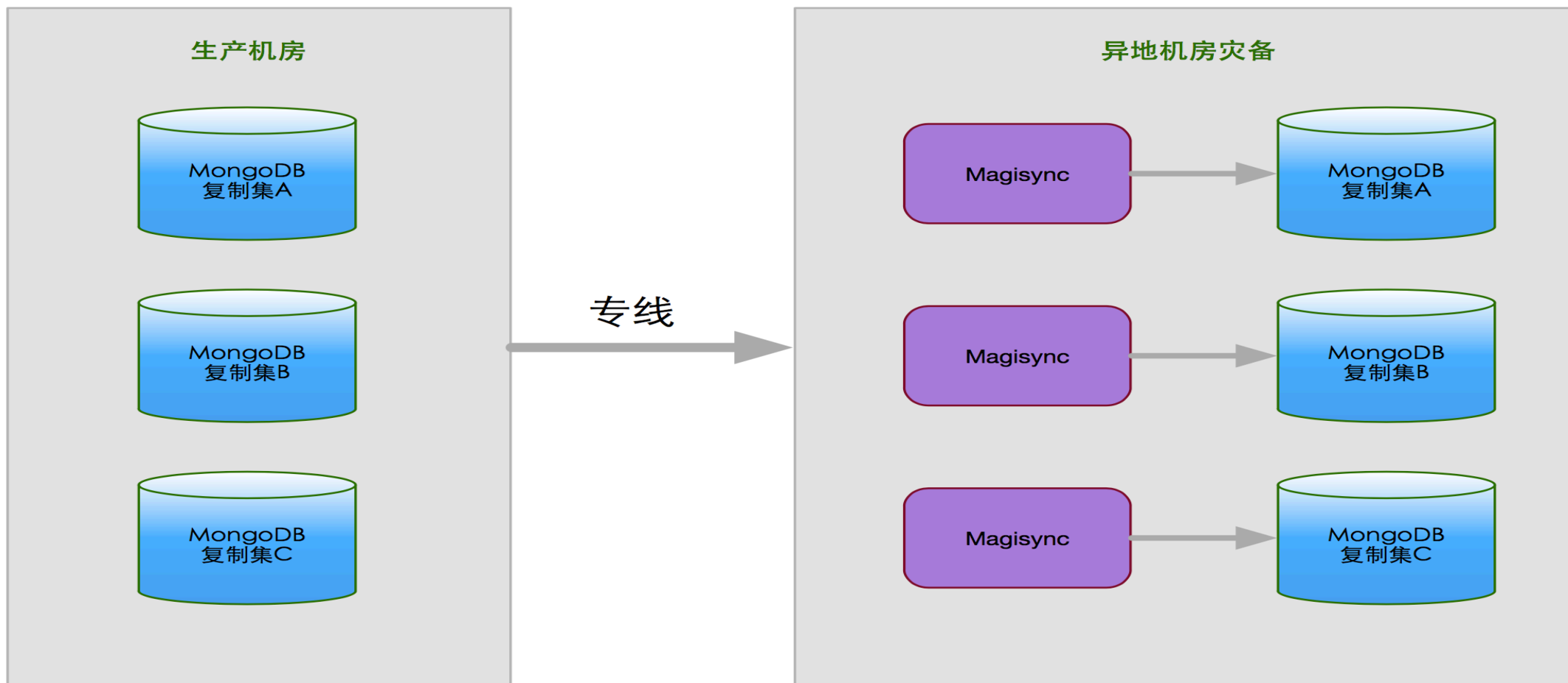
2015年8月到2016年8月
交易量增长了100倍

20150801 20150901 20151001 20151101 20151201 20160101 20160201 20160301 20160401 20160501 20160601 20160701 20160801 20160901 20161001

从系统架构来看 Magisync



从系统架构来看 Magisync



Magisync 同步特性



- ❖ 支持从特定时间点开始同步
- ❖ 支持选择同步特定表的特定类型的操作
- ❖ 支持断点续传式同步
- ❖ QPS 3000左右
- ❖ 支持限流

Magisync 同步原理



简单来说，Magisync 一直监听 MongoDB 的日志集合 oplog，如果源数据库有新的操作产生，oplog 集合中会生成有新的记录，这时，Magisync 会立刻拿到 oplog 中新的日志记录，并将其实时的同步至目标数据库。

Oplog

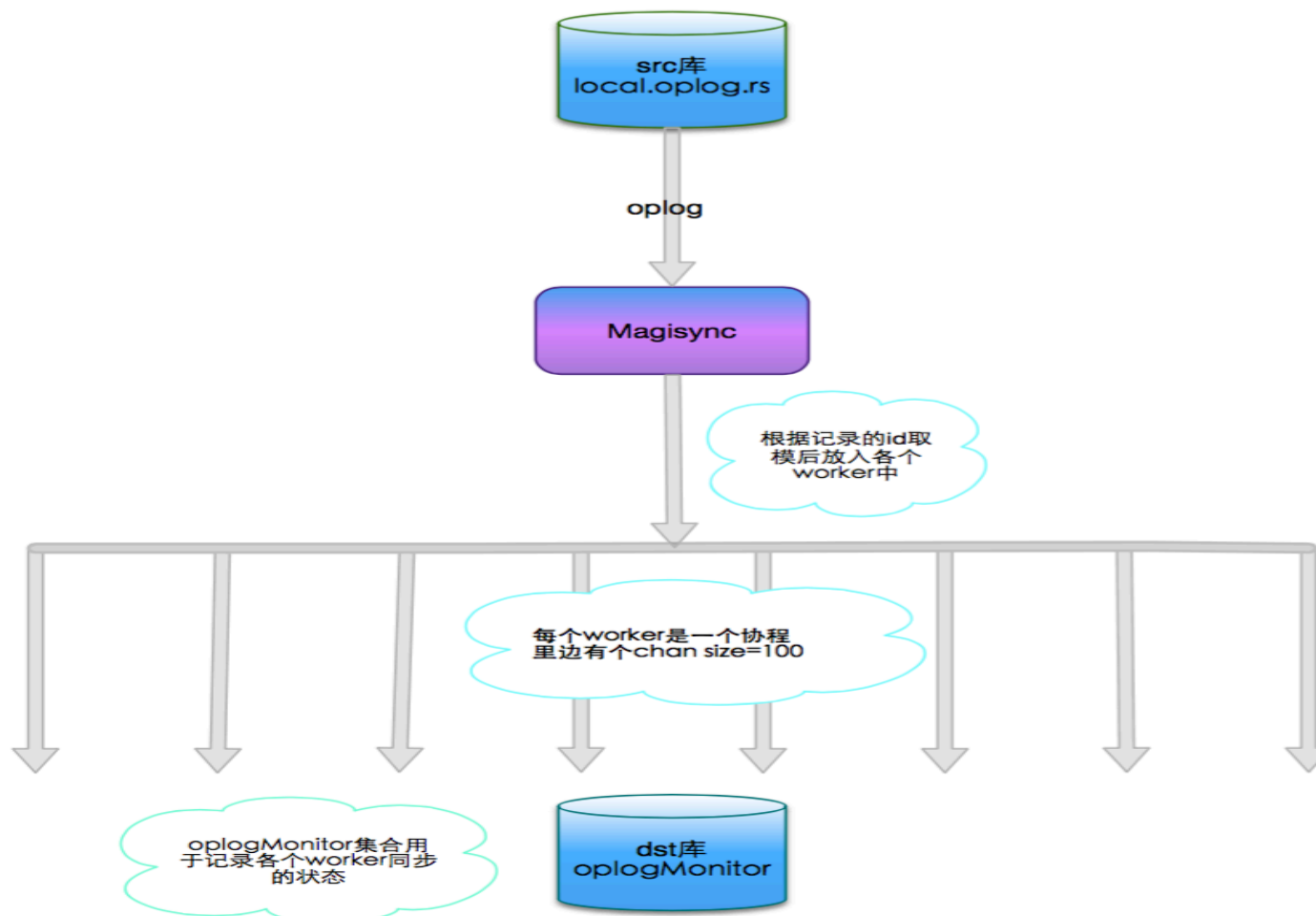
- ❖ MongoDB 的复制集存储写操作的一个日志
- ❖ Capped Collection
- ❖ 可以用类似于 Unix 中的 `tail -f`命令来获取 oplog
- ❖ 复制集中位置: `local.oplog.rs`

Oplog 的格式

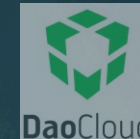
不同版本的 oplog 的格式稍有不同，我们生产环境使用的 MongoDB 的版本是3.0.3, oplog 示例如下：

```
{
  "ts" : Timestamp(1477194593, 1),
  "t" : NumberLong(12),
  "h" : NumberLong(8657529515144482533),
  "v" : NumberInt(2),
  "op" : "i",
  "ns" : "carolddb.user",
  "o" : {
    "_id" : ObjectId("580c333e7afa56085f89a00d"),
    "name" : "mike",
    "age" : NumberInt(20)
  }
}
```

Magisync 的实现原理



Magisync 特性说明



1 Magisync 如果重启，如何知道该从哪个时间点同步？

Magisync 各个协程去往目标数据同步之后，会将同步的状态写入 oplogMonitor 表中，每个协程在 oplogMonitor 表中都有一条对应的记录。等到 Magisync 重启时，则会读取 oplogMonitor 中最小的 ts，然后去源数据库拉取 oplog。

2 Magisync 用多协程去执行 oplog，如何保证 oplog 执行不会错序？

主线程读取到 oplog 后，会解析出 ObjectId，取模后将其放入各个 worker 对应的 chan 中。同一条数据，插入，更新等操作会进去到同一个 worker 中。

Magisync 特性说明



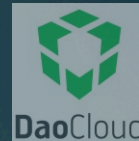
3 Magisync 的故障处理机制?

如果读取 oplog 失败，Magisync 会重连，重新读取 oplog；如果操作目标数据库失败，Magisync 会根据情况判断是否需要重连，或者重试。

4 Magisync 如何限流?

限流采用的是令牌桶的算法，会有一个协程定时的（比如每秒）向一个桶里放一定数量的令牌。主线程每读取一条 oplog，计算 oplog 的大小，从桶里消费对应的令牌数，然后对比桶里剩下的令牌数，如果小于0，则阻塞，直到下一秒桶里被放入令牌，阻塞解除。

Docker 化实践



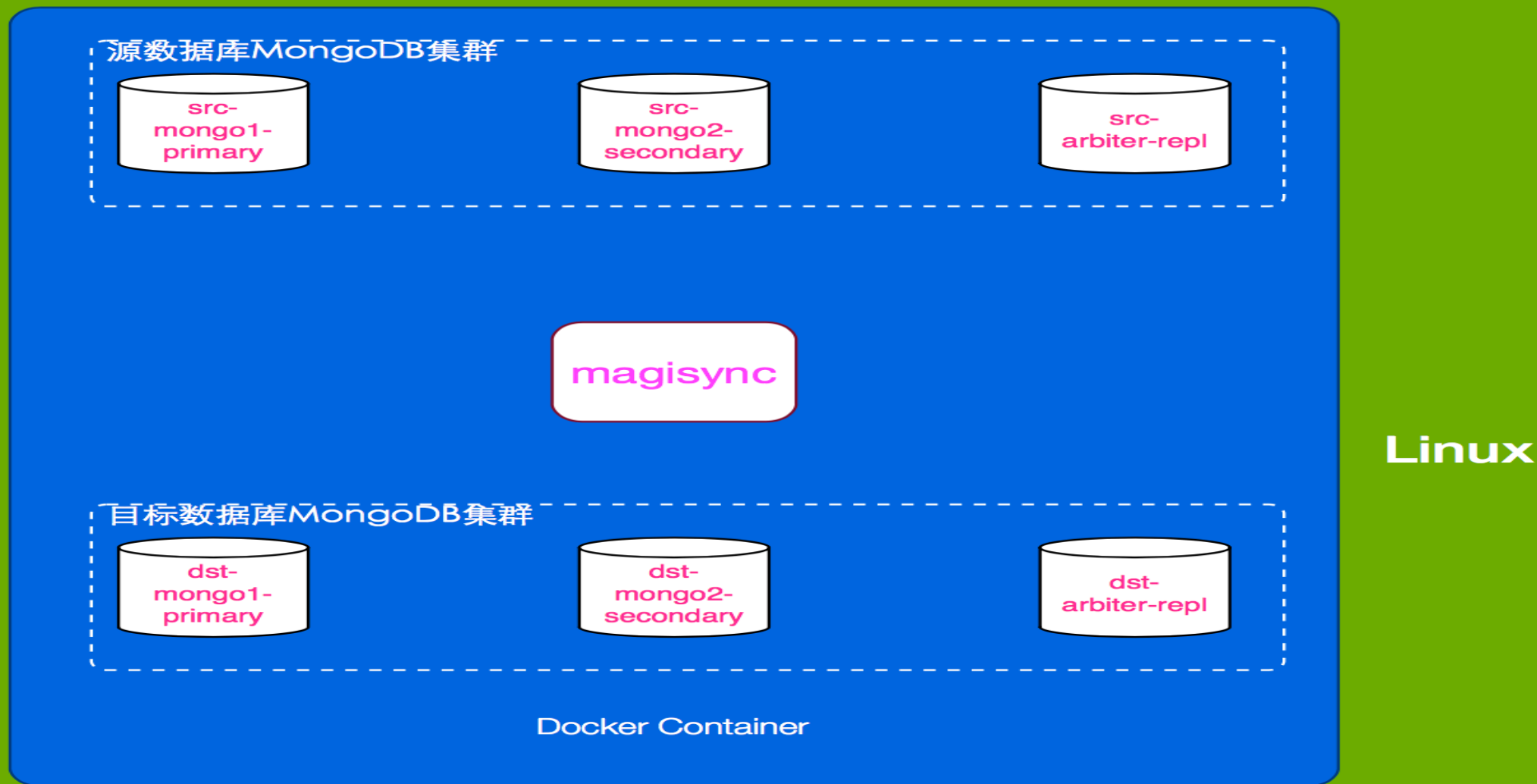
IT大咖说
不止于技术

为什么要使用Docker?

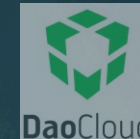
- ❖ 它真的很火。。。
- ❖ 快速构建应用环境
- ❖ 可移植性高



Docker 化实践



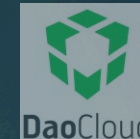
Magisync 代码示例



1 读取 oplog 代码

```
iter := p.srcSession.DB("local").C("oplog.rs").Find(q).Tail(-1)
for {
    var oplog bson.M
    if iter.Next(&oplog) {
        if goconf.Config.Oplog.IsLimit {
            p.bucket.Limit(oplog)
        }
        if !isProcessOplog(oplog) {
            log.Debugf("discard:%s", utils.ParseOplog(oplog))
            continue
        }
    }
}
```

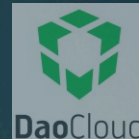
Magisync 代码示例



2 向目标数据库写数据

```
// replay oplog
func ReplayOplog(session *mongo.Session, oplog bson.M, dbname string) error {
    op := oplog["op"]
    ns := oplog["ns"]
    dbnameSrc := strings.Split(ns.(string), ".")[0]
    switch op {
    case "i": // insert
        collname := ns.(string)[len(dbnameSrc)+1:]
        if err := session.DB(dbname).C(collname).Insert(oplog["o"]); err != nil {
            log.Errorf("%s,%d,%s", utils.ParseOplog(oplog), 1, err)
            return err
        }
    }
}
```


Magisync 代码示例



```
case "u": // update
    collname := ns.(string)[len(dbnameSrc)+1:]
    if err := session.DB(dbname).C(collname).Update(oplog["o2"], oplog["o"]); err != nil {
        log.Errorf("%s,%d,%s", utils.ParseOplog(oplog), 1, err)
        return err
    }
case "d": // delete
    collname := ns.(string)[len(dbnameSrc)+1:]
    if err := session.DB(dbname).C(collname).Remove(oplog["o"]); err != nil {
        log.Errorf("%s,%d,%s", utils.ParseOplog(oplog), 1, err)
        return err
    }
case "c": // command
    if err := session.DB(dbname).Run(oplog["o"], nil); err != nil {
        log.Errorf("%s,%d,%s", utils.ParseOplog(oplog), 1, err)
        return err
    }
case "n": // no-op
    // do nothing
```

Magisync 开源计划



首先，Magisync 最初是参考了 GitHub 上的一个开源项目。地址为：
<https://github.com/caosiyang/go-mongo-sync>。

在这里感谢开源，让我们得到技术的共享。

Magisync 计划在2017年上半年开源。

谢谢大家！