Netkiller NoSQL 手札

目录

        工具
    7. MongoDB + Hadoop
    8. OSCM 一键安装 MongoDB 4.0.2
    9. Replication
    10. Drivers
        Using MongoDB in PHP
12. MongoDB 管理
    1. Security and Authentication
        超级管理员
        数据库访问用户
        数据库监控用户
        删除用户
        更新角色
    2. 4.0早期旧版本
13. 命令工具
    1. mongo - MongoDB Shell
        eval
        help
        登陆认证
        管道操作
    2. mongodump - Backup
        本地备份
        远程备份
    3. mongorestore
        本地恢复
        远程恢复
        filter
    4. mongostat
    5. mongotop
    6. mongofiles - Browse and modify a GridFS filesystem.
        list 浏览文件
        put 上传文件
        get 下载
        delete 删除
14. MongoDB Shell

1. shutdownServer
2. show 查看命令
    - show dbs
    - show collections
    - show users
    - show profile
3. 切换数据库
4. save
5. insert
6. update
    - multi 更新所有数据
    - upsert 更新，如果不存在则插入数据
7. remove
    - 删除条件使用 _id
8. 删除 collection
    - 删除字段
9. count()
10. 查询
    - find() MongoDB 2.x
    - find() MongoDB 3.x
        - Query
        - 包含字段
        - 排除字段
        - sort()
    - group()
11. aggregate
    - project
        - $split
        - substr
    - groupby + sum
12. Indexes 索引
    - 查看索引
    - 创建索引
    - 删除索引
    - 唯一索引
    - 复合索引

范例清单

Netkiller NoSQL 手札

MongoDB, Redis, CouchBase, Cassandra, Hypertable...

ISBN#

Mr. Neo Chan, 陈景峯(BG7NYT)

中国广东省深圳市望海路半岛城邦三期
518067
+86 13113668890

<netkiller@msn.com>

文档始创于2012-11-16
, $Date: 2013-04-25 16:24:49 +0800 (Thu, 25 Apr 2013) $

电子书最近一次更新于 2021-07-21 19:14:28

http://www.netkiller.cn
http://netkiller.github.io
http://netkiller.sourceforge.net
微信订阅 号 netkiller-ebook
微信：13113668890 请注明“读者”
QQ：13721218 请注明“读者”
QQ群：128659835 请注明“读者”

2017-02-13

Netkiller 数据库系列手札

数据库系列手札

Netkiller Database 手札 | Netkiller PostgreSQL 手札 | Netkiller MySQL 手札 | Netkiller NoSQL 手札 | Netkiller LDAP 手札

致读者

Netkiller 系列电子书始于 2000 年，风风雨雨走过20年，将在 2020 年终结，之后不在更新。作出这种决定原因 很多，例如现在的阅 读习惯已 经转向短视频，我个人的时间，身体健康情况等等.....

感谢读者粉丝这20年的支持

虽然电子书不再更新，后面我还会活跃在知乎社区和微信公众号

自述

《Netkiller 系列 手札》是一套免费系列电子书，netkiller 是nickname 从1999 开使用至今，“手札” 是札记，手册的含义。

2003年之前我还是以文章形式在BBS上发表各类技术文章，后来发现文章不够系统，便尝试写长篇技术文章加上章节目录等等。随着内容增加，不断修订，开始发布第一版，第二版.....

IT知识变化非常快，而且具有时效性，这样发布非常混乱，经常有读者发现第一版例子已经过时，但他不知道我已经发布第二版。

我便有一种想法，始终维护一个文档，不断更新，使他保持较新的版本不过时。

第一部电子书是《PostgreSQL 实用实例参考》开始我使用Microsoft Office Word 慢慢随着文档尺寸增加 Word 开始表现出力不从心。

我看到PostgreSQL 中文手册使用SGML编写文档，便开始学习Docbook SGML。使用Docbook写的第一部电子书是《Netkiller Postfix Integrated Solution》这是Netkiller 系列手札的原型。

至于“手札”一词的来历，是因为我爱好摄影，经常去一个台湾摄影网站，名字就叫“摄影家手札”。

由于硬盘损坏数据丢失 《Netkiller Postfix Integrated Solution》 的 SGML文件已经不存在; Docbook SGML存在很多缺陷UTF-8支持不好，转而使用Docbook XML.

目前技术书籍的价格一路飙升，动则￥80，￥100，少则￥50，￥60.技术书籍有时效性，随着技术的革新或淘汰，大批书记成为废纸垃

圾。并且这些书技术内容雷同，相互抄袭，质量越来越差，甚至里面给出的例子错误百出，只能购买影印版，或者翻译的版本。

在这种背景下我便萌生了自己写书的想法，资料主要来源是我的笔记与例子。我并不想出版，只为分享，所有我制作了基于CC License 发行的系列电子书。

本书注重例子，少理论（捞干货），只要你对着例子一步一步操作，就会成功，会让你有成就感并能坚持学下去，因为很多人遇到障碍就会放弃，其实我就是这种人，只要让他看到希望，就能坚持下去。

1.写给读者

为什么写这篇文章

有很多想法,工作中也用不到所以未能实现，所以想写出来,和大家分享.有一点写一点,写得也不好,只要能看懂就行,就当学习笔记了.

开始零零碎碎写过一些文档，也向维基百科供过稿，但维基经常被ZF封锁，后来发现sf.net可以提供主机存放文档，便做了迁移。并开始了我的写作生涯。

这篇文档是作者20年来对工作的总结,是作者一点一滴的积累起来的，有些笔记已经丢失，所以并不完整。

因为工作太忙整理比较缓慢。目前的工作涉及面比较窄所以新文档比较少。

我现在花在技术上的时间越来越少，兴趣转向摄影，无线电。也想写写摄影方面的心得体会。

写作动力:

曾经在网上看到外国开源界对中国的评价，中国人对开源索取无度，但贡献却微乎其微.这句话一直记在我心中，发誓要为中国开源事业做我仅有的一点微薄贡献

另外写文档也是知识积累，还可以增加在圈内的影响力.

人跟动物的不同,就是人类可以把自己学习的经验教给下一代人.下一代在上一代的基础上再创新,不断积累才有今天.

所以我把自己的经验写出来,可以让经验传承

没有内容的章节:

目前我自己一人维护所有文档，写作时间有限，当我发现一个好主题就会加入到文档中，待我有时间再完善章节，所以你会发现很多章节是空无内容的.

文档目前几乎是流水帐试的写作，维护量很大，先将就着看吧.

我想到哪写到哪,你会发现文章没一个中心,今天这里写点,明天跳过本章写其它的.
文中例子绝对多,对喜欢复制然后粘贴朋友很有用,不用动手写,也省时间.
理论的东西,网上大把,我这里就不写了,需要可以去网上查.
我爱写错别字,还有一些是打错的,如果发现请指正.
文中大部分试验是在Debian/Ubuntu/Redhat AS上完成.

写给读者

至读者:

我不知道什么时候，我不再更新文档或者退出IT行业去从事其他工作，我必须给这些文档找一个归宿，让他能持续更新下去。

我想捐赠给某些基金会继续运转，或者建立一个团队维护它。

我用了20年时间坚持不停地写作，持续更新，才有今天你看到的《Netkiller 手扎》系列文档，在中国能坚持20年，同时没有任何收益的技术类文档，是非常不容易的。

有很多时候想放弃，看到外国读者的支持与国内社区的影响，我坚持了下来。

中国开源事业需要各位参与，不要成为局外人，不要让外国人说:

中国对开源索取无度，贡献却微乎其微。
我们参与内核的开发还比较遥远，但是进个人能力，写一些文档还是可能的。

系列文档

下面是我多年积累下来的经验总结，整理成文档供大家参考:

Netkiller Architect 手札
Netkiller Developer 手札
Netkiller PHP 手札
Netkiller Python 手札
Netkiller Testing 手札
Netkiller Cryptography 手札
Netkiller Linux 手札
Netkiller FreeBSD 手札
Netkiller Shell 手札
Netkiller Security 手札
Netkiller Web 手札
Netkiller Monitoring 手札
Netkiller Storage 手札
Netkiller Mail 手札
Netkiller Docbook 手札
Netkiller Version 手札
Netkiller Database 手札
Netkiller PostgreSQL 手札
Netkiller MySQL 手札
Netkiller NoSQL 手札
Netkiller LDAP 手札
Netkiller Network 手札
Netkiller Cisco IOS 手札

Netkiller H3C 手札
Netkiller Multimedia 手札
Netkiller Management 手札
Netkiller Spring 手札
Netkiller Perl 手札
Netkiller Amateur Radio 手札

2. 作者简介

陈景峯 (ㄔㄣˊ ㄐㄧㄥˇ ㄈㄥ)

Nickname： netkiller | English name: Neo chen | Nippon name: ちん けいほう (音訳) | Korean name: 천징봉 | Thailand name: ภูมิภาพภูเขา | Vietnam: Trần Cảnh Phong

Callsign: BG7NYT | QTH: ZONE CQ24 ITU44 ShenZhen, China

程序猿，攻城狮，挨踢民工, Full Stack Developer, UNIX like Evangelist, 业余无线电爱好者（呼号：BG7NYT）,户外运动，山地骑行以及摄影爱好者。

《Netkiller 系列 手札》的作者

成长阶段

1981年1月19日(庚申年腊月十四)出生于黑龙江省青冈县建设乡双富大队第一小队

1989年9岁随父母迁居至黑龙江省伊春市，悲剧的天朝教育，不知道那门子归定，转学必须降一级，我本应该上一年级，但体制让我上学前班，那年多都10岁了

1995年小学毕业，体制规定借读要交3000两银子(我曾想过不升初中)，亲戚单位分楼告别平房，楼里没有地方放东西，把2麻袋书送给我，无意中发现一本电脑书BASIC语言，我竟然看懂了，对于电脑知识追求一发而不可收，后面顶零花钱，压岁钱主要用来买电脑书《MSDOS 6.22》《新编Unix实用大全》《跟我学Foxbase》。。。。。。

1996年第一次接触UNIX操作系统，BSD UNIX, Microsoft Xinux(盖茨亲自写的微软Unix，知道的人不多)

1997年自学Turbo C语言，苦于没有电脑，后来学校建了微机室才第一次使用QBASIC(DOS 6.22 自带命令)，那个年代只能通过软盘拷贝转播，Trubo C编译器始终没有搞到，

1997年第一次上Internet网速只有9600Bps, 当时全国兴起各种信息港域名格式是www.xxxx.info.net, 访问的第一个网站是NASA下载了很多火星探路者拍回的照片，还有“淞沪”sohu的前身

1998~2000年在哈尔滨学习计算机，充足的上机时间，但老师让我们练打字（明伦五笔/WT）打字不超过80个/每分钟还要强化训练，不过这个给我的键盘功夫打了好底。

1999年学校的电脑终于安装了光驱，在一张工具盘上终于找到了Turbo C, Borland C++与Quick Basic编译器，当时对VGA图形编程非常感兴趣，通过INT33中断控制鼠标，使用绘图函数模仿windows界面。还有操作 UCDOS 中文字库，绘制矢量与点阵字体。

2000年沉迷于Windows NT与Back Office各种技术，神马主域控制器，DHCP，WINS，IIS，域名服务器，Exchange邮件服务器，MS Proxy, NetMeeting...以及ASP+MS SQL开发；用56K猫下载了一张LINUX。ISO镜像，安装后我兴奋的24小时没有睡觉。

职业生涯

2001 年来深圳进城打工,成为一名外来务工者. 在一个4人公司做PHP开发，当时PHP的版本是2.0, 开始使用Linux Redhat 6.2.当时很多门户网站都是用FreeBSD,但很难搞到安装盘，在网易社区认识了一个网友,从广州给我寄了一张光盘，FreeBSD 3.2

2002 年我发现不能埋头苦干,还要学会"做人".后辗转广州工作了半年，考了一个Cisco CCNA认证。回到深圳重新开始，在车公庙找到一家工作做Java开发

2003 年这年最惨,公司拖欠工资16000元,打过两次官司2005才付清.

2004 年开始加入分布式计算团队,目前成绩，工作仍然是Java开发并且开始使用PostgreSQL数据库。

2004-10月开始玩户外和摄影

2005-6月成为中国无线电运动协会会员,呼号BG7NYT,进了一部Yaesu FT-60R手台。公司的需要转回PHP与MySQL，相隔几年发现PHP进步很大。在前台展现方面无人能敌，于是便前台使用PHP，后台采用Java开发。

2006 年单身生活了这么多年,终于找到归宿. 工作更多是研究PHP各种框架原理

2007 物价上涨,金融危机，休息了4个月（其实是找不到工作），关外很难上439.460中继，搞了一台Yaesu FT-7800.

2008 终于找到英文学习方法， 《Netkiller Developer 手札》，《Netkiller Document 手札》

2008-8-8 08:08:08 结婚,后全家迁居湖南省常德市

2009《Netkiller Database 手札》,2009-6-13学车，年底拿到C1驾照

2010 对电子打击乐产生兴趣，计划学习爵士鼓。由于我对Linux热爱，我轻松的接管了公司的运维部，然后开发运维两把抓。我印象最深刻的是公司一次上架10个机柜，我们用买服务器纸箱的钱改善伙食。我将40多台服务器安装BOINC做压力测试，获得了中国第二的名次。

2011 平凡的一年，户外运动停止，电台很少开，中继很少上，摄影主要是拍女儿与家人，年末买了一辆山地车

2012 对油笔画产生了兴趣，活动基本是骑行银湖山绿道，

2013 开始学习民谣吉他，同时对电吉他也极有兴趣；最终都放弃了。这一年深圳开始推数字中继2013-7-6日入手Motorola

MOTOTRBO XIR P8668，Netkiller 系列手札从Sourceforge向Github迁移；年底对MYSQL UDF，Engine与PHP扩展开发产生很浓的兴趣，拾起遗忘10+年的C，写了几个mysql扩展（图片处理，fifo管道与ZeroMQ），10月份入Toyota Rezi 2.5V并写了一篇《攻城狮的苦逼选车经历》

2014-9-8 在淘宝上买了一架电钢琴 Casio Privia PX-5S pro 开始陪女儿学习钢琴，由于这家钢琴是合成器电钢，里面有打击乐，我有对键盘鼓产生了兴趣。

2014-10-2号罗浮山两日游，对中国道教文化与音乐产生了兴趣，10月5号用了半天时间学会了简谱。10月8号入Canon 5D Mark III + Canon Speedlite 600EX-RT香港过关被查。

2014-12-20号对乐谱制作产生兴趣（https://github.com/SheetMusic/Piano），给女儿做了几首钢琴伴奏曲，MuseScore制谱然后生成MIDI与WAV文件。

2015-09-01 晚饭后拿起爵士鼓基础教程尝试在Casio Privia PX-5S pro演练，经过反复琢磨加上之前学钢琴的乐理知识，终于在02号晚上，打出了简单的基本节奏，迈出了第一步。

2016 对弓箭（复合弓）产生兴趣，无奈天朝法律法规不让玩。每周游泳轻松1500米无压力，年底入 xbox one s 和 Yaesu FT-2DR, 同时开始关注功放音响这块

2017 7月9号入 Yamaha RX-V581 功放一台，连接Xbox打游戏爽翻了，入Kindle电子书，计划学习蝶泳，果断放弃运维和开发知识体系转攻区块链。

2018 从溪山美地搬到半岛城邦，丢弃了多年攒下的家底。11 月开始玩 MMDVM，使用 Yaesu FT-7800 发射，连接MMDVM中继板，树莓派，覆盖深圳湾，散步骑车通联两不误。

2019 卖了常德的房子，住了5次院，哮喘反复发作，决定停止电子书更新，兴趣转到知乎，B站

2020 准备找工作

职业生涯路上继续打怪升级

3. 如何获得文档

下载 Netkiller 手札 (epub,kindle,chm,pdf)
EPUB https://github.com/netkiller/netkiller.github.io/tree/master/download/epub
MOBI https://github.com/netkiller/netkiller.github.io/tree/master/download/mobi
PDF https://github.com/netkiller/netkiller.github.io/tree/master/download/pdf
CHM https://github.com/netkiller/netkiller.github.io/tree/master/download/chm

通过 GIT 镜像整个网站

https://github.com/netkiller/netkiller.github.com.git

$ git clone https://github.com/netkiller/netkiller.github.com.git

镜像下载

整站下载

```
wget -m http://www.netkiller.cn/index.html
```

指定下载

```
wget -m wget -m http://www.netkiller.cn/linux/index.html
```

Yum 下载文档

获得光盘介质，RPM包，DEB包，如有特别需要，请联系我

YUM 在线安装电子书

http://netkiller.sourceforge.net/pub/repo/

```
# cat >> /etc/yum.repos.d/netkiller.repo <<EOF
[netkiller]
```

```
name=Netkiller Free Books
baseurl=http://netkiller.sourceforge.net/pub/repo/
enabled=1
gpgcheck=0
gpgkey=
EOF
```

查找包

```
# yum search netkiller

netkiller-centos.x86_64 : Netkiller centos Cookbook
netkiller-cryptography.x86_64 : Netkiller cryptography Cookbook
netkiller-docbook.x86_64 : Netkiller docbook Cookbook
netkiller-linux.x86_64 : Netkiller linux Cookbook
netkiller-mysql.x86_64 : Netkiller mysql Cookbook
netkiller-php.x86_64 : Netkiller php Cookbook
netkiller-postgresql.x86_64 : Netkiller postgresql Cookbook
netkiller-python.x86_64 : Netkiller python Cookbook
netkiller-version.x86_64 : Netkiller version Cookbook
```

安装包

```
yum install netkiller-docbook
```

4. 打赏(Donations)

If you like this documents, please make a donation to support the authors' efforts. Thank you!

您可以通过微信，支付宝，贝宝给作者打赏。

银行(Bank)

招商银行(China Merchants Bank)

开户名：陈景峰

账号：9555500000007459

微信(Wechat)

支付宝(Alipay)

PayPal Donations

https://www.paypal.me/netkiller

5. 联系方式

主站 http://www.netkiller.cn/

备用 http://netkiller.github.io/

繁体网站 http://netkiller.sourceforge.net/

联系作者

Mobile: +86 13113668890

Email: netkiller@msn.com

QQ群: 128659835 请注明“读者”

QQ: 13721218

ICQ: 101888222

注：请不要问我安装问题!

博客 **Blogger**

知乎专栏 https://zhuanlan.zhihu.com/netkiller

LinkedIn: http://cn.linkedin.com/in/netkiller

OSChina: http://my.oschina.net/neochen/

Facebook: https://www.facebook.com/bg7nyt

Flickr: http://www.flickr.com/photos/bg7nyt/

Disqus: http://disqus.com/netkiller/

solidot: http://solidot.org/~netkiller/

SegmentFault: https://segmentfault.com/u/netkiller

Reddit: https://www.reddit.com/user/netkiller/

Digg: http://www.digg.com/netkiller

Twitter: http://twitter.com/bg7nyt

weibo: http://weibo.com/bg7nyt

Xbox club

我的 xbox 上的ID是 netkiller xbox，我创建了一个俱乐部 netkiller 欢迎加入。

Radio

CQ CQ CQ DE BG7NYT:

如果这篇文章对你有所帮助,请寄给我一张QSL卡片, qrz.cn or qrz.com or hamcall.net

Personal Amateur Radiostations of P.R.China

ZONE CQ24 ITU44 ShenZhen, China

Best Regards, VY 73! OP. BG7NYT

守听频率 DMR 438.460 -8 Color 12 Slot 2 Group 46001

守听频率 C4FM 439.360 -5 DN/VW

MMDVM Hotspot:

Callsign: BG7NYT QTH: Shenzhen, China

YSF: YSF80337 - CN China 1 - W24166/TG46001

DMR: BM_China_46001 - DMR Radio ID: 4600441

部分 I. Redis

http://redis.io/

第 1 章 Redis 安装

1. CentOS 8 Stream

```
[root@netkiller ~]# dnf install -y redis
[root@netkiller ~]# systemctl enable redis
```

```
cp /etc/redis.conf{,.original}
sed -i 's/bind 127.0.0.1/bind 0.0.0.0/g' /etc/redis.conf
sed -i 's/timeout 0/timeout 30/' /etc/redis.conf
sed -i 's/# maxclients 10000/maxclients 1000/' /etc/redis.conf
sed -i 's/# maxmemory-policy noeviction/maxmemory-policy
volatile-lru/g' /usr/local/etc/redis.conf

random=$(cat /dev/urandom | tr -cd [:alnum:] | fold -w32 | head
-n 1)
sed -i "s/# requirepass foobared/requirepass ${random}/g"
/etc/redis.conf
echo "Redis random password is: ${random}"

cat >>  /etc/security/limits.d/20-nofile.conf <<EOF

redis soft nofile 65500
redis hard nofile 65500
EOF


cat >> /etc/sysctl.conf <<EOF
# Set up for Redis
vm.overcommit_memory = 1
net.core.somaxconn = 1024
EOF

sysctl -w net.core.somaxconn=1024
sysctl -w vm.overcommit_memory=1
```

```
cat >> /etc/rc.local  <<EOF

# Set up for Redis
echo never > /sys/kernel/mm/transparent_hugepage/enabled
EOF

echo never > /sys/kernel/mm/transparent_hugepage/enabled

systemctl enable redis
systemctl start redis
systemctl status redis
```

2. CentOS 7

Netkiller OSCM 一键安装

```
curl -s
https://raw.githubusercontent.com/oscm/shell/master/database/re
dis/redis.sh | bash
```

3. CentOS 6

安装fedora的YUM源，

```
rpm -Uvh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-7.noarch.rpm
```

安装redis

```
# yum install redis

# chkconfig redis on

# service redis start
```

备份配置文件，

```
# cp /etc/redis.conf /etc/redis.conf.original
```

3.1. 主从同步

主从同步配置非常简单，只需在从服务器 /etc/redis.conf 文件中开启 slaveof 即可

```
slaveof 192.168.2.1 6379
```

查看 /var/log/redis/redis.log 日志，可以看到同步情况

```
[20274] 09 Jul 13:13:53 * Server started, Redis version 2.4.10
```

```
[20274] 09 Jul 13:13:53 * DB loaded from disk: 0 seconds
[20274] 09 Jul 13:13:53 * The server is now ready to accept
connections on port 6379
[20274] 09 Jul 13:13:54 * Connecting to MASTER...
[20274] 09 Jul 13:13:54 * MASTER <-> SLAVE sync started
[20274] 09 Jul 13:13:54 * Non blocking connect for SYNC fired
the event.
[20274] 09 Jul 13:13:54 * MASTER <-> SLAVE sync: receiving 672
bytes from master
[20274] 09 Jul 13:13:54 * MASTER <-> SLAVE sync: Loading DB in
memory
[20274] 09 Jul 13:13:54 * MASTER <-> SLAVE sync: Finished with
success
```

3.2. Sentinel

4. Ubuntu

```
$ sudo apt-get install redis-server
```

```
$ dpkg -s redis-server
Package: redis-server
Status: install ok installed
Priority: optional
Section: database
Installed-Size: 208
Maintainer: Chris Lamb <lamby@debian.org>
Architecture: amd64
Source: redis
Version: 2:1.2.6-1
Depends: libc6 (>= 2.7), adduser
Conffiles:
 /etc/redis/redis.conf a19bad63017ec19def2c3a8a07bdc362
 /etc/logrotate.d/redis-server 06755b99ef70d62a56cff94cbfc36de7
 /etc/init.d/redis-server 3742555c10ab16fdd67fcbaf92faf694
 /etc/bash_completion.d/redis-cli
848565df7f222dc03c8d5cb34b9e0188
Description: Persistent key-value database with network
interface
 Redis is a key-value database in a similar vein to memcache
but the dataset
 is non-volatile. Redis additionally provides native support
for atomically
 manipulating and querying data structures such as lists and
sets.
 .
 The dataset is stored entirely in memory and periodically
flushed to disk.
Homepage: http://code.google.com/p/redis/
```

```
$ cat /etc/redis/redis.conf
```

```
# Redis configuration file example

# By default Redis does not run as a daemon. Use 'yes' if you
need it.
# Note that Redis will write a pid file in /var/run/redis.pid
when daemonized.
daemonize yes

# When run as a daemon, Redis write a pid file in
/var/run/redis.pid by default.
# You can specify a custom pid file location here.
pidfile /var/run/redis.pid

# Accept connections on the specified port, default is 6379
port 6379

# If you want you can bind a single interface, if the bind
option is not
# specified all the interfaces will listen for connections.
#
bind 127.0.0.1

# Close the connection after a client is idle for N seconds (0
to disable)
timeout 300

# Set server verbosity to 'debug'
# it can be one of:
# debug (a lot of information, useful for development/testing)
# notice (moderately verbose, what you want in production
probably)
# warning (only very important / critical messages are logged)
loglevel notice

# Specify the log file name. Also 'stdout' can be used to force
# the demon to log on the standard output. Note that if you use
standard
# output for logging but daemonize, logs will be sent to
/dev/null
logfile /var/log/redis/redis-server.log

# Set the number of databases. The default database is DB 0,
you can select
# a different one on a per-connection basis using SELECT <dbid>
```

```
where
# dbid is a number between 0 and 'databases'-1
databases 16

################################ SNAPSHOTTING
################################
#
# Save the DB on disk:
#
#   save <seconds> <changes>
#
#   Will save the DB if both the given number of seconds and
the given
#   number of write operations against the DB occurred.
#
#   In the example below the behaviour will be to save:
#   after 900 sec (15 min) if at least 1 key changed
#   after 300 sec (5 min) if at least 10 keys changed
#   after 60 sec if at least 10000 keys changed
save 900 1
save 300 10
save 60 10000

# Compress string objects using LZF when dump .rdb databases?
# For default that's set to 'yes' as it's almost always a win.
# If you want to save some CPU in the saving child set it to
'no' but
# the dataset will likely be bigger if you have compressible
values or keys.
rdbcompression yes

# The filename where to dump the DB
dbfilename dump.rdb

# For default save/load DB in/from the working directory
# Note that you must specify a directory not a file name.
dir /var/lib/redis

################################ REPLICATION
################################

# Master-Slave replication. Use slaveof to make a Redis
instance a copy of
# another Redis server. Note that the configuration is local to
the slave
```

```
# so for example it is possible to configure the slave to save
the DB with a
# different interval, or to listen to another port, and so on.
#
# slaveof <masterip> <masterport>

# If the master is password protected (using the "requirepass"
configuration
# directive below) it is possible to tell the slave to
authenticate before
# starting the replication synchronization process, otherwise
the master will
# refuse the slave request.
#
# masterauth <master-password>

################################## SECURITY
###################################

# Require clients to issue AUTH <PASSWORD> before processing
any other
# commands.  This might be useful in environments in which you
do not trust
# others with access to the host running redis-server.
#
# This should stay commented out for backward compatibility and
because most
# people do not need auth (e.g. they run their own servers).
#
# requirepass foobared

################################### LIMITS
####################################

# Set the max number of connected clients at the same time. By
default there
# is no limit, and it's up to the number of file descriptors
the Redis process
# is able to open. The special value '0' means no limts.
# Once the limit is reached Redis will close all the new
connections sending
# an error 'max number of clients reached'.
#
# maxclients 128
```

```
# Don't use more memory than the specified amount of bytes.
# When the memory limit is reached Redis will try to remove
keys with an
# EXPIRE set. It will try to start freeing keys that are going
to expire
# in little time and preserve keys with a longer time to live.
# Redis will also try to remove objects from free lists if
possible.
#
# If all this fails, Redis will start to reply with errors to
commands
# that will use more memory, like SET, LPUSH, and so on, and
will continue
# to reply to most read-only commands like GET.
#
# WARNING: maxmemory can be a good idea mainly if you want to
use Redis as a
# 'state' server or cache, not as a real DB. When Redis is used
as a real
# database the memory usage will grow over the weeks, it will
be obvious if
# it is going to use too much memory in the long run, and
you'll have the time
# to upgrade. With maxmemory after the limit is reached you'll
start to get
# errors for write operations, and this may even lead to DB
inconsistency.
#
# maxmemory <bytes>

############################## APPEND ONLY MODE
###############################

# By default Redis asynchronously dumps the dataset on disk. If
you can live
# with the idea that the latest records will be lost if
something like a crash
# happens this is the preferred way to run Redis. If instead
you care a lot
# about your data and don't want to that a single record can
get lost you should
# enable the append only mode: when this mode is enabled Redis
will append
# every write operation received in the file appendonly.log.
This file will
```

```
# be read on startup in order to rebuild the full dataset in
memory.
#
# Note that you can have both the async dumps and the append
only file if you
# like (you have to comment the "save" statements above to
disable the dumps).
# Still if append only mode is enabled Redis will load the data
from the
# log file at startup ignoring the dump.rdb file.
#
# The name of the append only file is "appendonly.log"
#
# IMPORTANT: Check the BGREWRITEAOF to check how to rewrite the
append
# log file in background when it gets too big.

appendonly no

# The fsync() call tells the Operating System to actually write
data on disk
# instead to wait for more data in the output buffer. Some OS
will really flush
# data on disk, some other OS will just try to do it ASAP.
#
# Redis supports three different modes:
#
# no: don't fsync, just let the OS flush the data when it
wants. Faster.
# always: fsync after every write to the append only log .
Slow, Safest.
# everysec: fsync only if one second passed since the last
fsync. Compromise.
#
# The default is "always" that's the safer of the options. It's
up to you to
# understand if you can relax this to "everysec" that will
fsync every second
# or to "no" that will let the operating system flush the
output buffer when
# it want, for better performances (but if you can live with
the idea of
# some data loss consider the default persistence mode that's
snapshotting).
```

```
appendfsync always
# appendfsync everysec
# appendfsync no

################################ ADVANCED CONFIG
################################

# Glue small output buffers together in order to send small
replies in a
# single TCP packet. Uses a bit more CPU but most of the times
it is a win
# in terms of number of queries per second. Use 'yes' if
unsure.
glueoutputbuf yes

# Use object sharing. Can save a lot of memory if you have many
common
# string in your dataset, but performs lookups against the
shared objects
# pool so it uses more CPU and can be a bit slower. Usually
it's a good
# idea.
#
# When object sharing is enabled (shareobjects yes) you can use
# shareobjectspoolsize to control the size of the pool used in
order to try
# object sharing. A bigger pool size will lead to better
sharing capabilities.
# In general you want this value to be at least the double of
the number of
# very common strings you have in your dataset.
#
# WARNING: object sharing is experimental, don't enable this
feature
# in production before of Redis 1.0-stable. Still please try
this feature in
# your development environment so that we can test it better.
shareobjects no
shareobjectspoolsize 1024
```

```
$ sudo /etc/init.d/redis-server start
```

5. Mac 安装 Redis

```
neo@MacBook-Pro ~ % brew install redis
```

Redis 被安装在 /usr/local/Cellar/redis/5.0.4 目录下

启动 Redis

```
brew services start redis
```

前台运行，这种方式用于调试

```
redis-server /usr/local/etc/redis.conf
```

6. 源码编译安装

这里仍然使用 Netkiller OSCM 安装，源码安装

```
curl -s
https://raw.githubusercontent.com/oscm/shell/master/database/redis/source/redis-5.0.4.sh | bash
```

7.

8. Test Redis

http://redis.io/commands

```
$ redis-cli info
redis_version:1.2.6
arch_bits:64
multiplexing_api:epoll
uptime_in_seconds:859
uptime_in_days:0
connected_clients:1
connected_slaves:0
used_memory:619490
used_memory_human:604.97K
changes_since_last_save:0
bgsave_in_progress:0
last_save_time:1311100746
bgrewriteaof_in_progress:0
total_connections_received:4
total_commands_processed:0
role:master

$ redis-cli set name neo
OK
$ redis-cli get name
neo

$ telnet localhost 6379
Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost (127.0.0.1).
Escape character is '^]'.
get name
$3
neo
quit
Connection closed by foreign host.
```

第 2 章 /etc/redis.conf

参数说明, redis.conf 配置项说明如下:

```
1. Redis默认不是以守护进程的方式运行，可以通过该配置项修改，使用yes启用守护进程
    daemonize no
2. 当Redis以守护进程方式运行时，Redis默认会把pid写入/var/run/redis.pid文件，可以通过pidfile指定
    pidfile /var/run/redis.pid
3. 指定Redis监听端口，默认端口为6379，作者在自己的一篇博文中解释了为什么选用6379作为默认端口，因为6379在手机按键上MERZ对应的号码，而MERZ取自意大利歌女Alessia Merz的名字
    port 6379
4. 绑定的主机地址
    bind 127.0.0.1
5.当 客户端闲置多长时间后关闭连接，如果指定为0，表示关闭该功能
    timeout 300
6. 指定日志记录级别，Redis总共支持四个级别：debug、verbose、notice、warning，默认为verbose
    loglevel verbose
7. 日志记录方式，默认为标准输出，如果配置Redis为守护进程方式运行，而这里又配置为日志记录方式为标准输出，则日志将会发送给/dev/null
    logfile stdout
8. 设置数据库的数量，默认数据库为0，可以使用SELECT <dbid>命令在连接上指定数据库id
    databases 16
9. 指定在多长时间内，有多少次更新操作，就将数据同步到数据文件，可以多个条件配合
    save <seconds> <changes>
    Redis默认配置文件中提供了三个条件：
    save 900 1
    save 300 10
    save 60 10000
    分别表示900秒（15分钟）内有1个更改，300秒（5分钟）内有10个更改以及60秒内有10000个更改。

10. 指定存储至本地数据库时是否压缩数据，默认为yes，Redis采用LZF压缩，如果为了节省CPU时间，可以关闭该选项，但会导致数据库文件变的巨大
    rdbcompression yes
```

11. 指定本地数据库文件名，默认值为dump.rdb
    dbfilename dump.rdb
12. 指定本地数据库存放目录
    dir ./
13. 设置当本机为slav服务时，设置master服务的IP地址及端口，在Redis启动时，它会自动从master进行数据同步
    slaveof <masterip> <masterport>
14. 当master服务设置了密码保护时，slav服务连接master的密码
    masterauth <master-password>
15. 设置Redis连接密码，如果配置了连接密码，客户端在连接Redis时需要通过AUTH <password>命令提供密码，默认关闭
    requirepass foobared
16. 设置同一时间最大客户端连接数，默认无限制，Redis可以同时打开的客户端连接数为Redis进程可以打开的最大文件描述符数，如果设置 maxclients 0，表示不作限制。当客户端连接数到达限制时，Redis会关闭新的连接并向客户端返回max number of clients reached错误信息
    maxclients 128
17. 指定Redis最大内存限制，Redis在启动时会把数据加载到内存中，达到最大内存后，Redis会先尝试清除已到期或即将到期的Key，当此方法处理 后，仍然到达最大内存设置，将无法再进行写入操作，但仍然可以进行读取操作。Redis新的vm机制，会把Key存放内存，Value会存放在swap区
    maxmemory <bytes>
18. 指定是否在每次更新操作后进行日志记录，Redis在默认情况下是异步的把数据写入磁盘，如果不开启，可能会在断电时导致一段时间内的数据丢失。因为 redis本身同步数据文件是按上面save条件来同步的，所以有的数据会在一段时间内只存在于内存中。默认为no
    appendonly no
19. 指定更新日志文件名，默认为appendonly.aof
     appendfilename appendonly.aof
20. 指定更新日志条件，共有3个可选值：
    no：表示等操作系统进行数据缓存同步到磁盘（快）
    always：表示每次更新操作后手动调用fsync()将数据写到磁盘（慢，安全）
    everysec：表示每秒同步一次（折衷，默认值）
    appendfsync everysec

21. 指定是否启用虚拟内存机制，默认值为no，简单的介绍一下，VM机制将数据分页存放，由Redis将访问量较少的页即冷数据swap到磁盘上，访问多的页面由磁盘自动换出到内存中（在后面的文章我会仔细分析Redis的VM机制）
     vm-enabled no
22. 虚拟内存文件路径，默认值为/tmp/redis.swap，不可多个Redis实例共享
     vm-swap-file /tmp/redis.swap
23. 将所有大于vm-max-memory的数据存入虚拟内存,无论vm-max-memory设置多小,所有索引数据都是内存存储的(Redis的索引数据 就是keys),也就是说,当vm-max-memory设置为0的时候,其实是所有value都存在于磁盘。默认值为0
     vm-max-memory 0

```
24. Redis swap文件分成了很多的page，一个对象可以保存在多个page上面，但一个page上不能被多个对象共享，vm-page-size是要根据存储的 数据大小来设定的，作者建议如果存储很多小对象，page大小最好设置为32或者64bytes；如果存储很大大对象，则可以使用更大的page，如果不 确定，就使用默认值
     vm-page-size 32
25. 设置swap文件中的page数量，由于页表（一种表示页面空闲或使用的bitmap）是在放在内存中的，，在磁盘上每8个pages将消耗1byte的内存。
     vm-pages 134217728
26. 设置访问swap文件的线程数,最好不要超过机器的核数,如果设置为0,那么所有对swap文件的操作都是串行的，可能会造成比较长时间的延迟。默认值为4
     vm-max-threads 4
27. 设置在向客户端应答时，是否把较小的包合并为一个包发送，默认为开启
    glueoutputbuf yes
28. 指定在超过一定的数量或者最大的元素超过某一临界值时，采用一种特殊的哈希算法
    hash-max-zipmap-entries 64
    hash-max-zipmap-value 512
29. 指定是否激活重置哈希，默认为开启（后面在介绍Redis的哈希算法时具体介绍）
    activerehashing yes
30. 指定包含其它的配置文件，可以在同一主机上多个Redis实例之间使用同一份配置文件，而同时各个实例又拥有自己的特定配置文件
    include /path/to/local.conf
```

1.密码认证

打开 /etc/redis.conf 修改 requirepass 配置项

```
# vim /etc/redis.conf

requirepass test123
```

测试

```

# service redis restart
Stopping redis-server:                                     [
OK  ]
```

```
Starting redis-server:                                             [
OK  ]

# redis-cli
redis 127.0.0.1:6379> set h helloworld
(error) ERR operation not permitted
```

auth test123

```
redis 127.0.0.1:6379> auth test123
OK
redis 127.0.0.1:6379> set h helloworld
OK
redis 127.0.0.1:6379> get h
"helloworld"
redis 127.0.0.1:6379> exit
```

redis-cli -a 参数指定密码

```
# redis-cli -a test123
redis 127.0.0.1:6379> set h helloworld
OK
redis 127.0.0.1:6379> get h
"helloworld"
redis 127.0.0.1:6379> exit
```

通过 config 动态改变密码，无需重新启动 redis 进程

```
# redis-cli -a test123
redis 127.0.0.1:6379> config get requirepass
1) "requirepass"
2) "test123"
```

```
redis 127.0.0.1:6379> config set requirepass passabc
OK
redis 127.0.0.1:6379> auth passabc
OK
redis 127.0.0.1:6379> set h helloworld
OK
redis 127.0.0.1:6379> get h
"helloworld"
redis 127.0.0.1:6379> exit
```

注意：config 不能保存到配置文件

```
# grep requirepass /etc/redis.conf

# If the master is password protected (using the "requirepass"
configuration
# requirepass foobared
requirepass test123
```

master/slave 模式， master 有密码, slave 怎样配置?

```
masterauth password
```

2. maxmemory-policy TTL 过期策略配置

```
1、volatile-lru：只对设置了过期时间的key进行LRU（默认值）
2、allkeys-lru ：  删除lru算法的key
3、volatile-random：随机删除即将过期key
4、allkeys-random：随机删除
5、volatile-ttl ：  删除即将过期的
6、noeviction ：  永不过期，返回错误
```

第 3 章 redis-cli - Command-line client to redis-server

1. 命令参数

1.1. password

```
-a <password>      Password to use when connecting to the
server.

[root@netkiller conf.d]# redis-cli -a
hsM8NK8b71vFQKFOS55jbWJrA1TYgI4e
```

1.2. raw

--raw Use raw formatting for replies (default when STDOUT is not a tty).

有时显示这样的数据

```
[hadoop@netkiller ~]$ redis-cli
127.0.0.1:6379> ZREVRANGE FASTNEWS_DATA_STORE_KEY 0 1
1) "
{\"title\":\"\xe4\xb8\x8a\xe8\xaf\x81\xe7\xbb\xbc\xe6\x8c\x87\x
e5\x91\xa8\xe7\xba\xbf\xe4\xba\x94\xe8\xbf\x9e\xe9\x98\xb3\xef\
xbc\x8c\xe5\x88\x9b\xe4\xb8\x9a\xe6\x9d\xbf\xe8\xbf\x9e\xe7\xbb
\xad\xe4\xb8\xa4\xe5\x91\xa8\xe5\xa4\xa7\xe8\xb7\x8c\xef\xbc\x9
b
\xe4\xb8\x8a\xe8\xaf\x81\xe7\xbb\xbc\xe6\x8c\x87\xe5\x91\xa8\xe
4\xba\x94\xe6\x94\xb6\xe7\x9b\x98\xe8\xb7\x8c0.23%\xef\xbc\x8c\
xe4\xb8\x8b\xe8\xb7\x8c7.60\xe7\x82\xb9\xef\xbc\x8c\xe6\x8a\xa5
3237.98\xe7\x82\xb9\xef\xbc\x9b
\xe6\xb7\xb1\xe8\xaf\x81\xe6\x88\x90\xe6\x8c\x87\xe5\x91\xa8\xe
```

```
4\xba\x94\xe6\x94\xb6\xe7\x9b\x98\xe8\xb7\x8c0.03%\xef\xbc\x8c\
xe4\xb8\x8b\xe8\xb7\x8c3.50\xe7\x82\xb9\xef\xbc\x8c\xe6\x8a\xa5
10363.48\xe7\x82\xb9\xef\xbc\x9b
\xe6\xb2\xaa\xe6\xb7\xb1300\xe8\x82\xa1\xe6\x8c\x87\xe5\x91\xa8
\xe4\xba\x94\xe6\x94\xb6\xe7\x9b\x98\xe8\xb7\x8c0.52%\xef\xbc\x
8c\xe4\xb8\x8b\xe8\xb7\x8c19.49\xe7\x82\xb9\xef\xbc\x8c\xe6\x8a
\xa53728.39\xe7\x82\xb9\xef\xbc\x9b
\xe5\x88\x9b\xe4\xb8\x9a\xe6\x9d\xbf\xe6\x8c\x87\xe6\x95\xb0\xe
5\x91\xa8\xe4\xba\x94\xe6\x94\xb6\xe7\x9b\x98\xe6\xb6\xa80.08%\
xef\xbc\x8c\xe4\xb8\x8a\xe6\xb6\xa81.31\xe7\x82\xb9\xef\xbc\x8c
\xe6\x8a\xa51689.92\xe7\x82\xb9\\r\\n\"}"
2) "
{\"title\":\"\xe4\xb8\x8a\xe8\xaf\x81\xe7\xbb\xbc\xe6\x8c\x87\x
e5\x91\xa8\xe4\xba\x94\xe6\x94\xb6\xe7\x9b\x98\xe4\xb8\x8b\xe8\
xb7\x8c0.21%\xef\xbc\x8c\xe6\x8a\xa53237.98\xe7\x82\xb9\xef\xbc
\x9b\xe6\xb7\xb1\xe8\xaf\x81\xe6\x88\x90\xe6\x8c\x87\xe5\x91\xa
8\xe4\xba\x94\xe6\x94\xb6\xe7\x9b\x98\xe5\xbe\xae\xe8\xb7\x8c0.
02%\xef\xbc\x8c\xe6\x8a\xa510364.82\xe7\x82\xb9\xef\xbc\x9b\xe5
\x88\x9b\xe4\xb8\x9a\xe6\x9d\xbf\xe6\x8c\x87\xe6\x95\xb0\xe5\x9
1\xa8\xe4\xba\x94\xe6\x94\xb6\xe7\x9b\x98\xe5\xbe\xae\xe6\xb6\x
a80.09%\xef\xbc\x8c\xe6\x8a\xa51690.15\xe7\x82\xb9\"}"
127.0.0.1:6379>
```

加入 --raw 参数后可以显示可以阅读的数据

```
[hadoop@VM_3_2_centos ~]$ redis-cli --raw
127.0.0.1:6379> ZREVRANGE FASTNEWS_DATA_STORE_KEY 0 1
{"title":"上证综指周线五连阳，创业板连续两周大跌"}
{"title":"上证综指周五收盘下跌0.21%，报3237.98点"}
127.0.0.1:6379>
```

2. --latency Enter a special mode continuously sampling latency.

参数的功能是从客户端发出一条命令到客户端接受到该命令的反馈所用的最长响应时间

```
# redis-cli --latency -h 192.168.2.1
min: 1, max: 210, avg: 3.64 (13453 samples)
```

3. auth

认证密码

```
[root@netkiller ~]# redis-cli
127.0.0.1:6379> keys *
(error) NOAUTH Authentication required.
127.0.0.1:6379> auth hsM8NKb71vjbWJrA1TYgI4
OK
127.0.0.1:6379> keys *
1) "HK50(1605)"
2) "GBPUSD"
3) "USDCHF"
4) "SP500(1609)"
5) "NZDJPY"
6) "AUDNZD"
7) "EURGBP"
8) "CLN6"
9) "BU6"
```

4. MONITOR

```
$ redis-cli monitor
```

5. info

```
redis 127.0.0.1:6379> info
redis_version:2.4.10
redis_git_sha1:00000000
redis_git_dirty:0
arch_bits:64
multiplexing_api:epoll
gcc_version:4.4.6
process_id:29663
uptime_in_seconds:1189
uptime_in_days:0
lru_clock:1018411
used_cpu_sys:0.10
used_cpu_user:0.09
used_cpu_sys_children:0.01
used_cpu_user_children:0.00
connected_clients:1
connected_slaves:0
client_longest_output_list:0
client_biggest_input_buf:0
blocked_clients:0
used_memory:730664
used_memory_human:713.54K
used_memory_rss:7225344
used_memory_peak:730720
used_memory_peak_human:713.59K
mem_fragmentation_ratio:9.89
mem_allocator:jemalloc-2.2.5
loading:0
aof_enabled:0
changes_since_last_save:0
bgsave_in_progress:0
last_save_time:1373332622
bgrewriteaof_in_progress:0
total_connections_received:4
total_commands_processed:14
expired_keys:0
evicted_keys:0
keyspace_hits:3
keyspace_misses:0
```

```
pubsub_channels:0
pubsub_patterns:0
latest_fork_usec:744
vm_enabled:0
role:master
db0:keys=4,expires=0
redis 127.0.0.1:6379>
```

6. save/bgsave/lastsave

save/bgsave 保存持久化将数据，lastsave 查看相关信息

```
redis 127.0.0.1:6379> save
OK
redis 127.0.0.1:6379> bgsave
Background saving started
redis 127.0.0.1:6379> lastsave
(integer) 1373335757
```

7. config

```
CONFIG GET dir
CONFIG GET dbfilename

config set stop-writes-on-bgsave-error yes
CONFIG SET dir /tmp
CONFIG SET dbfilename temp.rdb
```

8. keys

查询所有key

```
172.18.52.15:6379> keys *
 1) "www.example.com:743f10d0f1dc569ed5893856e14c1fb7captcha"
 2) "www.example.com:d88e0b6c54a235763dd731bcc0914439captcha"
 3)
"www.example.com:17f9091cb44f3cc5bb411eb801f07be8member_login"
 4)
"www.example.com:10ff594fd42f4c81212020555cfb586amember_login_i
nput"
 5) "www.example.com:a759ba5232ce324d0e6ae8da9290beaecaptcha"
 6) "www.example.com:37c78410af02d66a542d15b9707f215bcaptcha"
 7)
"www.example.com:9f5070e217f4eac9a1d15f9b8dbe7148deposit_1_temp
_var"
 8) "www.example.com:6c1a13c9396df2c35613043923bfe338captcha"
 9) "www.example.com:b611080c0627154871ea0e1498793238captcha"
10)
"www.example.com:2792241f8d0f075528db2b50e0c9c684member_login"
```

查询指定key

```
172.18.50.15:6379> set name neo
OK
172.18.50.15:6379> keys name
1) "name"
```

9. 字符串 操作

9.1. set/get/del

```
172.18.52.15:6379> set name neo
OK
172.18.52.15:6379> get name
"neo"
172.18.52.15:6379> keys name
1) "name"
172.18.52.15:6379> del name
(integer) 1
172.18.52.15:6379> get name
(nil)
```

9.2. setnx

SETNX key value

当 key 不存在时将 key 的值设为 value，若给定的 key 已经存在，则 SETNX 不做任何动作。SETNX 是(SET if Not eXists) (如果不存在，则 SET)的简写。

```
返回值:
        设置成功，返回 1
        设置失败，返回 0

redis> EXISTS neo             # neo 不存在
(integer) 0

redis> SETNX neo "chen"           # neo 设置成功
(integer) 1

redis> SETNX neo "netkiller"  # 尝试覆盖 neo ，失败
```

```
(integer) 0

redis> GET neo                    # 没有被覆盖
"chen"
```

10. expire/ttl

EXPIRE 设置过期时间, TTL 可以查询过期时间倒计时。

```
172.18.52.165:6379> set name neo
OK
172.18.52.165:6379> ttl name
(integer) -1
172.18.52.165:6379> expire name 30
(integer) 1
172.18.52.165:6379> ttl name
(integer) 22
172.18.52.165:6379> ttl name
(integer) 9
172.18.52.165:6379> ttl name
(integer) -1
172.18.52.165:6379> get  name
(nil)
```

注意ttl返回-1有两种情况，一是没有设置过期时间，另一种是该key已经过期不存在。

11. 获取 key 类型

```
root@netkiller ~ % redis-cli
127.0.0.1:6379> TYPE "logstash:redis"
list
```

12. LIST 数据类型

获取 list 列表长度

```
127.0.0.1:6379> TYPE "logstash:redis"
list
127.0.0.1:6379> LLEN "logstash:redis"
(integer) 69
127.0.0.1:6379> lpop "logstash:redis"
127.0.0.1:6379>
127.0.0.1:6379> LLEN "logstash:redis"
(integer) 68
```

```
1. LPUSH/LPUSHX/LRANGE:
/> redis-cli    #在Shell提示符下启动redis客户端工具。
redis 127.0.0.1:6379> del queue:test
(integer) 1
#queue:test键并不存在，该命令会创建该键及与其关联的List，之后在将参数中的values从左到右依次插入。
redis 127.0.0.1:6379> lpush queue:test a b c d
(integer) 4
#取从位置0开始到位置2结束的3个元素。
redis 127.0.0.1:6379> lrange queue:test 0 2
1) "d"
2) "c"
3) "b"
#取链表中的全部元素，其中0表示第一个元素，-1表示最后一个元素。
redis 127.0.0.1:6379> lrange queue:test 0 -1
1) "d"
2) "c"
3) "b"
4) "a"
#queue:test2键此时并不存在，因此该命令将不会进行任何操作，其返回值为0。
redis 127.0.0.1:6379> lpushx queue:test2 e
(integer) 0
#可以看到queue:test2没有关联任何List Value。
```

```
redis 127.0.0.1:6379> lrange queue:test2 0 -1
(empty list or set)
#queue:test键此时已经存在，所以该命令插入成功，并返回链表中当前元素的数量。
redis 127.0.0.1:6379> lpushx queue:test e
(integer) 5
#获取该键的List Value的头部元素。
redis 127.0.0.1:6379> lrange queue:test 0 0
1) "e"

2. LPOP/LLEN:
redis 127.0.0.1:6379> lpush queue:test a b c d
(integer) 4
redis 127.0.0.1:6379> lpop queue:test
"d"
redis 127.0.0.1:6379> lpop queue:test
"c"
#在执行lpop命令两次后，链表头部的两个元素已经被弹出，此时链表中元素的数量是2
redis 127.0.0.1:6379> llen queue:test
(integer) 2

3. LREM/LSET/LINDEX/LTRIM:
#为后面的示例准备测试数据。
redis 127.0.0.1:6379> lpush queue:test a b c d a c
(integer) 6
#从头部(left)向尾部(right)变量链表，删除2个值等于a的元素，返回值为实际删除的数量。
redis 127.0.0.1:6379> lrem queue:test 2 a
(integer) 2
#看出删除后链表中的全部元素。
redis 127.0.0.1:6379> lrange queue:test 0 -1
1) "c"
2) "d"
3) "c"
4) "b"
#获取索引值为1(头部的第二个元素)的元素值。
redis 127.0.0.1:6379> lindex queue:test 1
"d"
#将索引值为1(头部的第二个元素)的元素值设置为新值e。
redis 127.0.0.1:6379> lset queue:test 1 e
OK
#查看是否设置成功。
redis 127.0.0.1:6379> lindex queue:test 1
"e"
```

```
#索引值6超过了链表中元素的数量，该命令返回nil。
redis 127.0.0.1:6379> lindex queue:test 6
(nil)
#设置的索引值6超过了链表中元素的数量，设置失败，该命令返回错误信息。
redis 127.0.0.1:6379> lset queue:test 6 hh
(error) ERR index out of range
#仅保留索引值0到2之间的3个元素，注意第0个和第2个元素均被保留。
redis 127.0.0.1:6379> ltrim queue:test 0 2
OK
#查看trim后的结果。
redis 127.0.0.1:6379> lrange queue:test 0 -1
1) "c"
2) "e"
3) "c"

4. LINSERT:
#删除该键便于后面的测试。
redis 127.0.0.1:6379> del queue:test
(integer) 1
#为后面的示例准备测试数据。
redis 127.0.0.1:6379> lpush queue:test a b c d e
(integer) 5
#在a的前面插入新元素a1。
redis 127.0.0.1:6379> linsert queue:test before a a1
(integer) 6
#查看是否插入成功，从结果看已经插入。注意lindex的index值是0-based。
redis 127.0.0.1:6379> lindex queue:test 0
"e"
#在e的后面插入新元素e2，从返回结果看已经插入成功。
redis 127.0.0.1:6379> linsert queue:test after e e2
(integer) 7
#再次查看是否插入成功。
redis 127.0.0.1:6379> lindex queue:test 1
"e2"
#在不存在的元素之前或之后插入新元素，该命令操作失败，并返回-1。
redis 127.0.0.1:6379> linsert queue:test after k a
(integer) -1
#为不存在的Key插入新元素，该命令操作失败，返回0。
redis 127.0.0.1:6379> linsert queue:test1 after a a2
(integer) 0

5. RPUSH/RPUSHX/RPOP/RPOPLPUSH:
#删除该键，以便于后面的测试。
redis 127.0.0.1:6379> del queue:test
(integer) 1
```

```
#从链表的尾部插入参数中给出的values，插入顺序是从左到右依次插入。
redis 127.0.0.1:6379> rpush queue:test a b c d
(integer) 4
#通过lrange的可以获悉rpush在插入多值时的插入顺序。
redis 127.0.0.1:6379> lrange queue:test 0 -1
1) "a"
2) "b"
3) "c"
4) "d"
#该键已经存在并且包含4个元素，rpushx命令将执行成功，并将元素e插入到链表的尾部。
redis 127.0.0.1:6379> rpushx queue:test e
(integer) 5
#通过lindex命令可以看出之前的rpushx命令确实执行成功，因为索引值为4的元素已经是新元素了。
redis 127.0.0.1:6379> lindex queue:test 4
"e"
#由于queue:test2键并不存在，因此该命令不会插入数据，其返回值为0。
redis 127.0.0.1:6379> rpushx queue:test2 e
(integer) 0
#在执行rpoplpush命令前，先看一下queue:test中链表的元素有哪些，注意他们的位置关系。
redis 127.0.0.1:6379> lrange queue:test 0 -1
1) "a"
2) "b"
3) "c"
4) "d"
5) "e"
#将queue:test的尾部元素e弹出，同时再插入到queue:test2的头部(原子性的完成这两步操作)。
redis 127.0.0.1:6379> rpoplpush queue:test queue:test2
"e"
#通过lrange命令查看queue:test在弹出尾部元素后的结果。
redis 127.0.0.1:6379> lrange queue:test 0 -1
1) "a"
2) "b"
3) "c"
4) "d"
#通过lrange命令查看queue:test2在插入元素后的结果。
redis 127.0.0.1:6379> lrange queue:test2 0 -1
1) "e"
#将source和destination设为同一键，将queue:test中的尾部元素移到其头部。
redis 127.0.0.1:6379> rpoplpush queue:test queue:test
"d"
#查看移动结果。
```

```
redis 127.0.0.1:6379> lrange queue:test 0 -1
1) "d"
2) "a"
3) "b"
4) "c"
```

13. set 无序字符集合

Set和List类型不同的是，Set集合中不允许出现重复的元素，和List类型相比，Set类型在功能上还存在着一个非常重要的特性，即在服务器端完成多个Sets之间的聚合计算操作，如unions、intersections和differences。由于这些操作均在服务端完成，因此效率极高，而且也节省了大量的网络IO开销。

1. SADD/SMEMBERS/SCARD/SISMEMBER:

```
    #插入测试数据，由于该键set:test之前并不存在，因此参数中的三个成员都被正常插入。
    redis 127.0.0.1:6379> sadd set:test a b c
    (integer) 3
    #由于参数中的a在set:test中已经存在，因此本次操作仅仅插入了d和e两个新成员。
    redis 127.0.0.1:6379> sadd set:test a d e
    (integer) 2
    #判断a是否已经存在，返回值为1表示存在。
    redis 127.0.0.1:6379> sismember set:test a
    (integer) 1
    #判断f是否已经存在，返回值为0表示不存在。
    redis 127.0.0.1:6379> sismember set:test f
    (integer) 0
    #通过smembers命令查看插入的结果，从结果可以，输出的顺序和插入顺序无关。
    redis 127.0.0.1:6379> smembers set:test
    1) "c"
    2) "d"
    3) "a"
    4) "b"
    5) "e"
    #获取Set集合中元素的数量。
    redis 127.0.0.1:6379> scard set:test
    (integer) 5
```

2. SPOP/SREM/SRANDMEMBER/SMOVE:

```
    #删除该键，便于后面的测试。
    redis 127.0.0.1:6379> del set:test
    (integer) 1
    #为后面的示例准备测试数据。
    redis 127.0.0.1:6379> sadd set:test a b c d
    (integer) 4
    #查看Set中成员的位置。
    redis 127.0.0.1:6379> smembers set:test
    1) "c"
    2) "d"
    3) "a"
    4) "b"
    #从结果可以看出，该命令确实是随机的返回了某一成员。
    redis 127.0.0.1:6379> srandmember set:test
    "c"
    #Set中尾部的成员b被移出并返回，事实上b并不是之前插入的第一个或最后一个成员。
    redis 127.0.0.1:6379> spop set:test
    "b"
    #查看移出后Set的成员信息。
    redis 127.0.0.1:6379> smembers set:test
    1) "c"
    2) "d"
    3) "a"
    #从Set中移出a、d和f三个成员，其中f并不存在，因此只有a和d两个成员被移出，返回为2。
    redis 127.0.0.1:6379> srem set:test a d f
    (integer) 2
    #查看移出后的输出结果。
    redis 127.0.0.1:6379> smembers set:test
    1) "c"
    #为后面的smove命令准备数据。
    redis 127.0.0.1:6379> sadd set:test a b
    (integer) 2
    redis 127.0.0.1:6379> sadd set:test2 c d
    (integer) 2
    #将a从set:test移到set:test2，从结果可以看出移动成功。
    redis 127.0.0.1:6379> smove set:test set:test2 a
    (integer) 1
    #再次将a从set:test移到set:test2，由于此时a已经不是set:test的成员了，因此移动失败并返回0。
    redis 127.0.0.1:6379> smove set:test set:test2 a
    (integer) 0
```

```
    #分别查看set:test和set:test2的成员，确认移动是否真的成功。
    redis 127.0.0.1:6379> smembers set:test
    1) "b"
    redis 127.0.0.1:6379> smembers set:test2
    1) "c"
    2) "d"
    3) "a"
```

3. SDIFF/SDIFFSTORE/SINTER/SINTERSTORE:

```
    #为后面的命令准备测试数据。
    redis 127.0.0.1:6379> sadd set:test a b c d
    (integer) 4
    redis 127.0.0.1:6379> sadd set:test2 c
    (integer) 1
    redis 127.0.0.1:6379> sadd set:test3 a c e
    (integer) 3
    #set:test和set:test2相比，a、b和d三个成员是两者之间的差异成员。再用这个结果继续和set:test3进行差异比较，b和d是set:test3不存在的成员。
    redis 127.0.0.1:6379> sdiff set:test set:test2 set:test3
    1) "d"
    2) "b"
    #将3个集合的差异成员存在在diffkey关联的Set中，并返回插入的成员数量。
    redis 127.0.0.1:6379> sdiffstore diffkey set:test set:test2 set:test3
    (integer) 2
    #查看一下sdiffstore的操作结果。
    redis 127.0.0.1:6379> smembers diffkey
    1) "d"
    2) "b"
    #从之前准备的数据就可以看出，这三个Set的成员交集只有c。
    redis 127.0.0.1:6379> sinter set:test set:test2 set:test3
    1) "c"
    #将3个集合中的交集成员存储到与interkey关联的Set中，并返回交集成员的数量。
    redis 127.0.0.1:6379> sinterstore interkey set:test set:test2 set:test3
    (integer) 1
    #查看一下sinterstore的操作结果。
    redis 127.0.0.1:6379> smembers interkey
    1) "c"
```

```
    #获取3个集合中的成员的并集。
    redis 127.0.0.1:6379> sunion set:test set:test2 set:test3
    1) "b"
    2) "c"
    3) "d"
    4) "e"
    5) "a"
    #将3个集合中成员的并集存储到unionkey关联的set中，并返回并集成员的数量。
    redis 127.0.0.1:6379> sunionstore unionkey set:test set:test2 set:test3
    (integer) 5
    #查看一下suiionstore的操作结果。
    redis 127.0.0.1:6379> smembers unionkey
    1) "b"
    2) "c"
    3) "d"
    4) "e"
    5) "a"
```

14. zset (有序集合)

添加到集合

```
root@netkiller ~ % redis-cli -n 16
127.0.0.1:6379[16]> zadd book 1 "Linux"
1
127.0.0.1:6379[16]> zadd book 2 "Java"
1
127.0.0.1:6379[16]> zadd book 3 "Python"
1
127.0.0.1:6379[16]> zadd book 4 "PHP"
1
127.0.0.1:6379[16]>
```

zrange 查看集合内 容

```
127.0.0.1:6379[16]> zrange book 0 -1 withscores
1) "Linux"
2) "1"
3) "Java"
4) "2"
5) "Perl"
6) "5"
```

指定开始和结束范围

```
127.0.0.1:6379[16]> zrange book 0 4
Linux
Java
Python
```

```
PHP

127.0.0.1:6379[16]> zrange book 1 4
Java
Python
PHP

127.0.0.1:6379[16]> zrange book 2 3
Python
PHP
```

zrem 删除集合成员

```


127.0.0.1:6379[16]> zadd book 5 "Rabby"
1
127.0.0.1:6379[16]> zrange book 4 5
Perl
Rabby
127.0.0.1:6379[16]> zrem book Rabby
1
127.0.0.1:6379[16]> zrange book 4 5
Perl

127.0.0.1:6379[16]> zrem book PHP Python
2
```

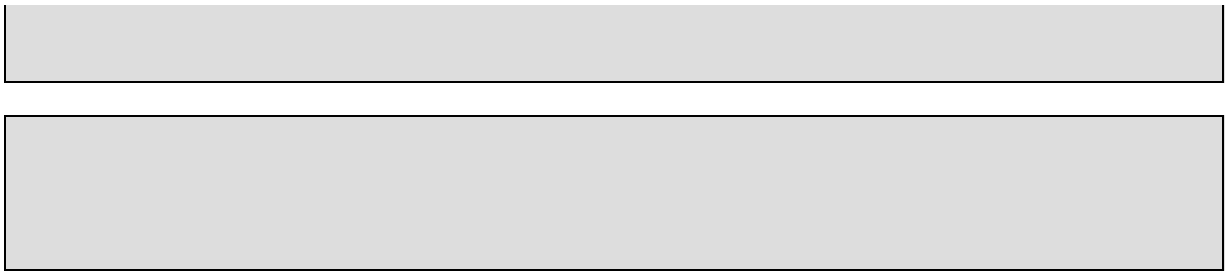
zcard 返回成员数量

```


127.0.0.1:6379[16]> zcard book
3
```

```

```



15. Pub/Sub 订阅与发布

redis 提供基本的MQ 功能，下面我们做一个演示

开启第一个终端窗口，订阅 first second

```
$ redis-cli
redis 127.0.0.1:6379> SUBSCRIBE first second
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "first"
3) (integer) 1
1) "subscribe"
2) "second"
3) (integer) 2
```

开启第二个终端窗口，分别发送first second

```
$ redis-cli
redis 127.0.0.1:6379> PUBLISH second Hello
(integer) 1
redis 127.0.0.1:6379> PUBLISH first Helloworld!!!
(integer) 1
redis 127.0.0.1:6379> quit
```

现在切换到第一个终端窗口，应该能够看到发送过来的字符串

```
$ redis-cli
redis 127.0.0.1:6379> SUBSCRIBE first second
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "first"
3) (integer) 1
1) "subscribe"
2) "second"
3) (integer) 2
```

```
1) "message"
2) "second"
3) "Hello"

1) "message"
2) "first"
3) "Helloworld!!!"
```

16. flushdb 清空 Redis 数据

```
root@netkiller ~ % redis-cli
127.0.0.1:6379> flushdb
OK
127.0.0.1:6379>
```

第 4 章 redis-benchmark 测试工具

redis-benchmark 基准性能测试

```
用法 redis-benchmark [-h <host>][-p ] [-c <clients>][-n ]> [-k
<boolean>]

选项:

选项      说明
-h <hostname>    主机名 （默认 127.0.0.1）
-p <port>        主机端口 （默认 6379）
-s <socket>      UNIX socket （会覆盖 -h -p 设置的内容）
-a <password>    密码（密码错误之类不会直接保错，而是在操作时才会保错，这
时可以使用 Redis 的 AUTH 命令再次认证）
-c <clients>     客户端的并发数量（默认是50）
-n <requests>    客户端请求总量（默认是100000）
-d <size>                使用 SET/GET 添加的数据的字节大小 (默认 2)
-dbnum <db>              选择一个数据库进行测试 (默认 0)
-k <boolean>     客户端是否使用keepalive，1为使用，0为不使用，（默认为
1）
-r <keyspacelen>使用 SET/GET/INCR 命令添加数据 key, SADD 添加随机数
据，keyspacelen 指定的是添加 键的数量
-P <numreq>              每个请求 pipeline 的数据量（默认为1，没有
pipeline ）
-q                               仅仅显示redis-benchmark的requests
per second信息
--csv                    将结果按照csv格式输出，便于后续处理
-l                               循环测试
-t <tests>               可以对指定命令进行基准测试
-I                               空闲模式 只打开N个空闲连接并等待。
```

代表256各个客户端同时请求 Redis，一 共执行 20000 次。redis-benchmark会对各类数据结构的命令进行测试，并给 出性能指标:

```
redis-benchmark -c 256 -n 20000
```

第 5 章 Redis Cluster

```
CLUSTER INFO 打印集群的信息
CLUSTER NODES 列出集群当前已知的所有节点（node），以及这些节点的相关信息。
CLUSTER RESET reset 重置
CLUSTER SAVECONFIG 强制节点保存集群当前状态到磁盘上。
CLUSTER SLOTS 获得slot在节点上的映射关系
CLUSTER MEET 将 ip 和 port 所指定的节点添加到集群当中，让它成为集群的一份子。
CLUSTER FORGET 从集群中移除 node_id 指定的节点。
CLUSTER REPLICATE 将当前节点设置为 node_id 指定的节点的从节点。
CLUSTER SLAVES 列出该slave节点的master节点
CLUSTER ADDSLOTS [slot ...] 将一个或多个槽（slot）指派（assign）给当前节点。
CLUSTER DELSLOTS [slot ...] 移除一个或多个槽对当前节点的指派。
CLUSTER FLUSHSLOTS 移除指派给当前节点的所有槽，让当前节点变成一个没有指派任何槽的节点。
CLUSTER SETSLOT NODE 将槽 slot 指派给 node_id 指定的节点，如果槽已经指派给另一个节点，那么先让另一个节点删除该槽>，然后再进行指派。
CLUSTER SETSLOT MIGRATING 将本节点的槽 slot 迁移到 node_id 指定的节点中。
CLUSTER SETSLOT IMPORTING 从 node_id 指定的节点中导入槽 slot 到本节点。
CLUSTER SETSLOT STABLE 取消对槽 slot 的导入（import）或者迁移
（migrate）。
CLUSTER KEYSLOT 计算键 key 应该被放置在哪个槽上。
CLUSTER COUNTKEYSINSLOT 返回槽 slot 目前包含的键值对数量。
CLUSTER GETKEYSINSLOT 返回 count 个 slot 槽中的键
READONLY 在集群中的salve节点开启只读模式
READWRITE 禁止读取请求跳转到集群中的salve节点上clust
```

第 6 章 Redis 通信协议

1. 切换DB

select n 切换DB， n表示数据库ID

```
# telnet 192.168.41.160 6379
Trying 192.168.41.160...
Connected to 192.168.41.160.
Escape character is '^]'.
select 1
+OK
```

2. 监控

telnet方式

```
# telnet 172.18.52.13 6379
Trying 172.18.52.163...
Connected to 172.18.52.13.
Escape character is '^]'.
MONITOR
+OK
+1425454378.190210 "MONITOR"
+1425454381.165317 "GET" "admin:633"
+1425454381.165725 "SET" "admin:633" "
{\"id\":\"633\",\"username\":\"7209\",\"password\":\"eea5981a4f
d021b8d78f8431084ba760\",\"status\":\"N\",\"belong_user_id\":\"
133\",\"level_id\":\"67\",\"create_time\":1425454381,\"session_
id\":\"l1s609t9gq8nj7vc94hb1i3s25\"}"
+1425454381.166088 "EXPIRE" "admin:633" "3600"
+1425454387.956387 "GET" "admin:633"
```

使用 nc 监控状态

```
# (echo -en "MONITOR\r\n"; sleep 10) | nc 172.18.52.13 6379
```

第 7 章 phpRedisAdmin

https://github.com/ErikDubbelboer/phpRedisAdmin

Example

You can find an example database at
http://dubbelboer.com/phpRedisAdmin/

第一种方法

```
git clone https://github.com/ErikDubbelboer/phpRedisAdmin
cd phpRedisAdmin
git clone https://github.com/nrk/predis
```

第二种方法

```
git clone https://github.com/ErikDubbelboer/phpRedisAdmin.git
cd phpRedisAdmin
git submodule init
git submodule update
```

第 8 章 Redis 开发

1.消息订阅与发布

订阅

```
<?php
$redis = new Redis();
$redis->connect('127.0.0.1',6379);
$channel = $argv[1];  // channel
$redis->subscribe(array('channel'.$channel), 'callback');
function callback($instance, $channelName, $message) {
  echo $channelName, "==>", $message,PHP_EOL;
}
```

发布

```
<?php
$redis = new Redis();
$redis->connect('127.0.0.1',6379);
$channel = $argv[1];  // channel
$msg = $argv[2];     // msg
$redis->publish('channel'.$channel, $msg);
```

第 9 章 A fast, light-weight proxy for memcached and redis

https://github.com/twitter/twemproxy

第 10 章 FAQ

1.清空数据库

```
FLUSHDB - Removes data from your connection's CURRENT database.
FLUSHALL - Removes data from ALL databases.
```

2. (error) MISCONF Redis is configured to save RDB snapshots

(error) MISCONF Redis is configured to save RDB snapshots, but it is currently not able to persist on disk. Commands that may modify the data set are disabled, because this instance is configured to report errors during writes if RDB snapshotting fails (stop-writes-on-bgsave-error option). Please check the Redis logs for details about the RDB error.

临时解决方案

```
# redis-cli

127.0.0.1:6379> config set stop-writes-on-bgsave-error no
OK
127.0.0.1:6379> set name neo
OK
127.0.0.1:6379> get name
"neo"
```

原因 是数据库持久化写入硬盘出现问题，可能是数据库目录权限不足。

排查方法

```
CONFIG GET dir
CONFIG GET dbfilename

config set stop-writes-on-bgsave-error yes
CONFIG SET dir /tmp
CONFIG SET dbfilename temp.rdb
```

尝试解决

```
# redis-cli
127.0.0.1:6379> CONFIG GET dir
1) "dir"
2) "/"
127.0.0.1:6379> CONFIG GET dbfilename
1) "dbfilename"
2) "dump.rdb"
127.0.0.1:6379> config set stop-writes-on-bgsave-error yes
OK
127.0.0.1:6379> CONFIG SET dir /tmp
OK
127.0.0.1:6379> set test aaaa
OK
127.0.0.1:6379> get test
"aaaa"
```

如果确认是 dir 权限 问题，我们就通过修改redis.conf 中得dir配置解决。

部分 II. MongoDB

http://www.mongodb.org/

1. FAQ

1.1. MongoDB 3.x 启用认证后恢复数据库需指定 collection

```
                        # mongorestore -u yourdb dump/
                        Enter password:

                        2017-06-09T11:55:58.566+0800 Failed:
error connecting to db server: server returned error on SASL
authentication step: Authentication failed.
```

```
                        # mongorestore -u yourdb -d yourdb
dump/yourdb
```

1.2. MongoDB 2.x 早期版本用户管理

```
> use admin
switched to db admin
> db.addUser('neo','chen')
{
        "user" : "neo",
        "readOnly" : false,
        "pwd" : "68ace374737253d87e0ec91d4fcb673d"
}

> db.system.users.find()
{ "_id" : ObjectId("4c481404b9db6474d2fcb76f"), "user" : "neo",
"readOnly" : false, "pwd" : "68ace374737253d87e0ec91d4fcb673d"
}
```

```
> db.auth('neo','chen')
1
```

1.3. Failed: netkiller.assets: error reading database: command listCollections requires authentication

```
[root@ecs-3705 ~]# mongorestore dump/
2018-11-05T11:48:08.981+0800    preparing collections to
restore from
2018-11-05T11:48:08.982+0800    Failed: netkiller.assets: error
reading database: command listCollections requires
authentication
```

需要认证，请使用 -u 用户名 -p 密码 -d 数据库 来恢复

```
[root@netkiller ~]# mongorestore -h 127.0.0.1 -u netkiller -p
netkiller -d netkiller  dump/netkiller/
```

第 11 章 Install 安装MongoDB

1. CentOS 8 Stream

```
#!/bin/sh

cat << 'EOF' >> /etc/yum.repos.d/mongodb-org-5.0.repo
[mongodb-org-5.0]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb
-org/5.0/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-5.0.asc
EOF

dnf install -y mongodb-org-server
dnf install -y mongodb-org-shell
dnf install -y mongodb-org-tools

cp /etc/mongod.conf{,.original}


cat << 'EOF' >>  /etc/security/limits.d/20-nofile.conf
mongod soft nofile 65000
mongod hard nofile 65000
EOF

systemctl is-enabled mongod
systemctl start mongod
```

2. MacOS 安装 MongoDB

```
brew install mongodb
```

启动

```
brew services start mongodb
```

3. 二进制tar包安装

Install MongoDB

```
wget http://fastdl.mongodb.org/linux/mongodb-linux-x86_64-
1.5.5.tgz

debian:/srv# tar zxf mongodb-linux-x86_64-1.5.5.tgz
debian:/srv# ln -s mongodb-linux-x86_64-1.5.5 mongodb
```

Create a data directory

By default MongoDB will store data in /data/db, but it won't automatically create that directory. To create it, do:

```
$ sudo mkdir -p /data/db/
$ sudo chown `id -u` /data/db
```

You can also tell MongoDB to use a different data directory, with the --dbpath option.

Run and connect to the server

First, start the MongoDB server in one terminal:

```
$ ./mongodb/bin/mongod
```

In a separate terminal, start the shell, which will connect to localhost by default:

```
$ ./mongodb/bin/mongo
> db.foo.save( { a : 1 } )
```

```
> db.foo.find()
```

Congratulations, you've just saved and retrieved your first document with MongoDB!

例 11.1. MongoDB Test

```
debian:/srv/mongodb/bin# ./mongo
MongoDB shell version: 1.5.5
connecting to: test
[initandlisten] Thu Jul 22 16:42:07 connection accepted from
127.0.0.1:42876 #1
> db.foo.save({a:1})
Thu Jul 22 16:42:23 allocating new datafile /data/db/test.ns,
filling with zeroes...
Thu Jul 22 16:42:23 done allocating datafile /data/db/test.ns,
size: 16MB,  took 0.025 secs
Thu Jul 22 16:42:23 allocating new datafile /data/db/test.0,
filling with zeroes...
Thu Jul 22 16:42:23 done allocating datafile /data/db/test.0,
size: 64MB,  took 0.105 secs
[conn1] Thu Jul 22 16:42:23 building new index on { _id: 1 }
for test.foo
[conn1] Thu Jul 22 16:42:23 Buildindex test.foo idxNo:0 { name:
"_id_", ns: "test.foo", key: { _id: 1 } }
[conn1] Thu Jul 22 16:42:23 done for 0 records 0secs
[conn1] Thu Jul 22 16:42:23 insert test.foo 136ms
> Thu Jul 22 16:42:23 allocating new datafile /data/db/test.1,
filling with zeroes...
Thu Jul 22 16:42:24 done allocating datafile /data/db/test.1,
size: 128MB,  took 0.228 secs
> db.foo.find()
{ "_id" : ObjectId("4c48046f74050cbf6c9a0ef6"), "a" : 1 }

> use neo
switched to db neo
> db.foo.save({a:1})
Thu Jul 22 16:54:50 allocating new datafile /data/db/neo.ns,
filling with zeroes...
Thu Jul 22 16:54:50 done allocating datafile /data/db/neo.ns,
size: 16MB,  took 0.026 secs
```

```
Thu Jul 22 16:54:50 allocating new datafile /data/db/neo.0,
filling with zeroes...
Thu Jul 22 16:54:50 done allocating datafile /data/db/neo.0,
size: 64MB,  took 0.122 secs
[conn1] Thu Jul 22 16:54:50 building new index on { _id: 1 }
for neo.foo
[conn1] Thu Jul 22 16:54:50 Buildindex neo.foo idxNo:0 { name:
"_id_", ns: "neo.foo", key: { _id: 1 } }
Thu Jul 22 16:54:50 allocating new datafile /data/db/neo.1,
filling with zeroes...
[conn1] Thu Jul 22 16:54:50 done for 0 records 0.01secs
[conn1] Thu Jul 22 16:54:50 insert neo.foo 164ms
> Thu Jul 22 16:54:50 done allocating datafile /data/db/neo.1,
size: 128MB,  took 0.217 secs

> db.foo.find()
{ "_id" : ObjectId("4c48075a74050cbf6c9a0ef7"), "a" : 1 }
>

> db.neo.save({a:1})
[conn1] Thu Jul 22 16:58:32 building new index on { _id: 1 }
for neo.neo
[conn1] Thu Jul 22 16:58:32 Buildindex neo.neo idxNo:0 { name:
"_id_", ns: "neo.neo", key: { _id: 1 } }
[conn1] Thu Jul 22 16:58:32 done for 0 records 0.024secs
> db.neo.find()
{ "_id" : ObjectId("4c48083874050cbf6c9a0ef8"), "a" : 1 }
```

Starting Mongo

Running as a Daemon

```
$ ./mongod --fork --logpath /var/log/mongodb.log --logappend
```

4. Ubuntu MongoDB

```
$ sudo apt-get install mongodb-server mongodb-clients
```

```
$ /etc/init.d/mongodb
Usage: /etc/init.d/mongodb {start|stop|force-
stop|restart|force-reload|status}
```

5. CentOS 7 MongoDB

CentOS 默认 MongoDB 是 2.6.12

```
[root@iZ623h9icu8Z ~]# yum info mongodb
Loaded plugins: langpacks
Repodata is over 2 weeks old. Install yum-cron? Or run: yum
makecache fast
Available Packages
Name        : mongodb
Arch        : x86_64
Version     : 2.6.12
Release     : 3.el7
Size        : 43 M
Repo        : epel/x86_64
Summary     : High-performance, schema-free document-oriented
database
URL         : http://www.mongodb.org
License     : AGPLv3 and zlib and ASL 2.0
Description : Mongo (from "humongous") is a high-performance,
open source, schema-free
            : document-oriented database. MongoDB is written in
C++ and offers the following
            : features:
            :     * Collection oriented storage: easy storage
of object/JSON-style data
            :     * Dynamic queries
            :     * Full index support, including on inner
objects and embedded arrays
            :     * Query profiling
            :     * Replication and fail-over support
            :     * Efficient storage of binary data including
large objects (e.g. photos
            :     and videos)
            :     * Auto-sharding for cloud-level scalability
(currently in early alpha)
            :     * Commercial Support Available
            :
            : A key goal of MongoDB is to bridge the gap
between key/value stores (which are
            : fast and highly scalable) and traditional RDBMS
systems (which are deep in
```

```
            : functionality).
```

```
# yum install mongodb-server

# chkconfig mongod on

# service mongod start
```

单独安装客户端

```
# yum install mongodb
```

6. 从官网安装最新版本的 MongoDB 3.4

官网的rpm包是如下

```
[root@netkiller ~]# yum search mongodb | grep "\-org"
mongodb-org.x86_64 : MongoDB open source document-oriented
database system
mongodb-org-mongos.x86_64 : MongoDB sharded cluster query router
mongodb-org-server.x86_64 : MongoDB database server
mongodb-org-shell.x86_64 : MongoDB shell client
mongodb-org-tools.x86_64 : MongoDB tools
```

```
#!/bin/sh
cat << 'EOF' >> /etc/yum.repos.d/mongodb-org-3.4.repo
[mongodb-org-3.4]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-
org/3.4/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-3.4.asc
EOF
```

Server

```
yum install -y mongodb-org-server

cp /etc/mongod.conf{,.original}

systemctl is-enabled mongod
systemctl start mongod
```

Client

```
yum install -y mongodb-org-shell
```

```
[root@netkiller ~]# mongo

MongoDB shell version v3.4.0
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.0
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2016-11-30T11:34:36.493+0800 I STORAGE  [initandlisten]
2016-11-30T11:34:36.560+0800 I CONTROL  [initandlisten]
2016-11-30T11:34:36.560+0800 I CONTROL  [initandlisten] **
WARNING: Access control is not enabled for the database.
2016-11-30T11:34:36.560+0800 I CONTROL  [initandlisten] **
Read and write access to data and configuration is unrestricted.
2016-11-30T11:34:36.560+0800 I CONTROL  [initandlisten]
2016-11-30T11:34:36.560+0800 I CONTROL  [initandlisten]
2016-11-30T11:34:36.560+0800 I CONTROL  [initandlisten] **
WARNING: /sys/kernel/mm/transparent_hugepage/enabled is
'always'.
2016-11-30T11:34:36.560+0800 I CONTROL  [initandlisten] **
We suggest setting it to 'never'
2016-11-30T11:34:36.560+0800 I CONTROL  [initandlisten]
2016-11-30T11:34:36.560+0800 I CONTROL  [initandlisten] **
WARNING: /sys/kernel/mm/transparent_hugepage/defrag is 'always'.
2016-11-30T11:34:36.560+0800 I CONTROL  [initandlisten] **
We suggest setting it to 'never'
2016-11-30T11:34:36.560+0800 I CONTROL  [initandlisten]
> show dbs
admin  0.000GB
local  0.000GB
> exit
bye
```

工具

```
# yum install mongodb-org-tools

# rpm -ql mongodb-org-tools.x86_64 0:3.4.1-1.el7
/usr/bin/bsondump
/usr/bin/mongodump
/usr/bin/mongoexport
/usr/bin/mongofiles
/usr/bin/mongoimport
/usr/bin/mongooplog
/usr/bin/mongoperf
/usr/bin/mongorestore
/usr/bin/mongostat
/usr/bin/mongotop
/usr/share/man/man1/bsondump.1
/usr/share/man/man1/mongodump.1
/usr/share/man/man1/mongoexport.1
/usr/share/man/man1/mongofiles.1
/usr/share/man/man1/mongoimport.1
/usr/share/man/man1/mongooplog.1
/usr/share/man/man1/mongoperf.1
/usr/share/man/man1/mongorestore.1
/usr/share/man/man1/mongostat.1
/usr/share/man/man1/mongotop.1
```

7. MongoDB + Hadoop

Hadoop Connector

http://docs.mongodb.org/ecosystem/tutorial/getting-started-with-hadoop/

```
git clone https://github.com/mongodb/mongo-hadoop.git
git checkout release-1.0
```

```
# vim build.sbt
hadoopRelease in ThisBuild := "cdh4"
```

```
./sbt package
```

```
wget --no-check-certificate
https://github.com/downloads/mongodb/mongo-java-driver/mongo-
2.7.3.jar
cp mongo-2.7.3.jar /usr/lib/hadoop/lib/
cp core/target/mongo-hadoop-core_cdh3u3-1.0.0.jar
/usr/lib/hadoop/lib/
```

待续.....

8. OSCM 一键安装 MongoDB 4.0.2

安装 MongoDB

```
curl -s
https://raw.githubusercontent.com/oscm/shell/master/database/mongodb/mon
godb.org/mongodb-4.0.2.sh | bash
```

创建管理和数据库用户

```
# mongo

use admin;
db.createUser(
   {
     user: "admin",
     pwd: "chen",
     roles: [ "dbAdmin", "dbOwner", "userAdmin" ]
   }
);

use products
db.createUser(
   {
     user: "accountUser",
     pwd: "password",
     roles: [ "readWrite", "dbAdmin" ]
   }
)
```

开启认证

```
curl -s
https://raw.githubusercontent.com/oscm/shell/master/database/mongodb/mon
godb.org/security.authorization.enabled.sh | bash
```

9. Replication

很多教程上面采用手工配置主从复制，我不建议你这样启动，请采用修改/etc/mongod.conf配置文件的方案。

```
创建主：

mongod –port 27017 –dbpath /var/lib/mongodb –master

 创建从：

mongod –port 27017 –dbpath /var/lib/mongodb –slave –source
master_ip_address:27017
                </screen>

                <section>
                        <title>Master</title>
                        <screen><![CDATA[
sed -i "s/#master = true/master = true/" /etc/mongod.conf

systemctl restart  mongod
                        </screen>
                </section>
                <section>
                        <title>Slave</title>
                        <screen><![CDATA[
sed -i "s/#slave = true/slave = true/" /etc/mongod.conf
sed -i "s/#source = arg/source = mongodb.master.example.com/"
/etc/mongod.conf

systemctl restart  mongod
                        </screen>
                </section>
                <section>
                        <title>测试</title>
                        <para>进入 Master</para>
                        <screen>
                        <![CDATA[
[root@localhost ~]# mongo
MongoDB shell version: 2.6.11
connecting to: test
Welcome to the MongoDB shell.
```

```
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2015-11-14T15:51:21.215+0800 [initandlisten]
2015-11-14T15:51:21.215+0800 [initandlisten] ** WARNING:
Readahead for /var/lib/mongodb is set to 4096KB
2015-11-14T15:51:21.215+0800 [initandlisten] **          We
suggest setting it to 256KB (512 sectors) or less
2015-11-14T15:51:21.215+0800 [initandlisten] **
http://dochub.mongodb.org/core/readahead
>
>
> db.foo.save({'name':'neo','address':
{'city':'shenzhen','post':518000},'phone':
[13113668890,13322993040]})
WriteResult({ "nInserted" : 1 })
> db.foo.find();
{ "_id" : ObjectId("5646e881a11081d5998bf70c"), "name" : "neo",
"address" : { "city" : "shenzhen", "post" : 518000 }, "phone" :
[ 13113668890, 13322993040 ] }
>
```

进入 Slave

```
[root@localhost ~]# mongo
MongoDB shell version: 2.6.11
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2015-11-14T15:51:23.668+0800 [initandlisten]
2015-11-14T15:51:23.668+0800 [initandlisten] ** WARNING:
Readahead for /var/lib/mongodb is set to 4096KB
2015-11-14T15:51:23.668+0800 [initandlisten] **          We
```

```
suggest setting it to 256KB (512 sectors) or less
2015-11-14T15:51:23.668+0800 [initandlisten] **
http://dochub.mongodb.org/core/readahead
> db.foo.find();
{ "_id" : ObjectId("5646e881a11081d5998bf70c"), "name" : "neo",
"address" : { "city" : "shenzhen", "post" : 518000 }, "phone" :
[ 13113668890, 13322993040 ] }
>
```

10. Drivers

Using MongoDB in PHP

Installing the PHP Driver

```
sudo pecl install mongo
```

Open your php.ini file and add to it:

```
extension=mongo.so
```

例 11.2. Using MongoDB in PHP

```
[root@subversion html]# cat mongo.php
<?php

// connect
$m = new Mongo('192.168.3.9');

// select a database
$db = $m->comedy;
$collection = $db->cartoons;

// add an element
$obj = array( "title" => "Calvin and Hobbes", "author" => "Bill
Watterson" );
$collection->insert($obj);

// add another element, with a different "shape"
$obj = array( "title" => "XKCD", "online" => true );
$collection->insert($obj);
```

```
// find everything in the collection
$cursor = $collection->find();

// iterate through the results
foreach ($cursor as $obj) {
    echo $obj["title"] . "\n";
}

// disconnect
$m->close();

?>
```

```
[root@subversion html]# php mongo.php
Calvin and Hobbes
XKCD
[root@subversion html]# php mongo.php
Calvin and Hobbes
XKCD
Calvin and Hobbes
XKCD
                                </screen>
                                <para></para>
                                <screen>
                                <![CDATA[
> use comedy
switched to db comedy
> db.foo.find()
> db.cartoons.find()
{ "_id" : ObjectId("4c481d2b9503c17611000000"), "title" :
"Calvin and Hobbes", "author" : "Bill Watterson" }
{ "_id" : ObjectId("4c481d2b9503c17611010000"), "title" :
"XKCD", "online" : true }
{ "_id" : ObjectId("4c481d2f9503c17711000000"), "title" :
"Calvin and Hobbes", "author" : "Bill Watterson" }
{ "_id" : ObjectId("4c481d2f9503c17711010000"), "title" :
"XKCD", "online" : true }
>
```

第 12 章 MongoDB 管理

1. Security and Authentication

```
sed -i "32s/#security:/security:/" /etc/mongod.conf
sed -i "33 i \ \ authorization: enabled" /etc/mongod.conf
```

超级管理员

Database Administration Roles

```
use admin
db.createUser(
  {
    user: "admin",
    pwd: passwordPrompt(), // or cleartext password
    roles: [ { role: "userAdminAnyDatabase", db: "admin" },
"readWriteAnyDatabase" ]
  }
)
```

数据库访问用户

注意，只有创建了超级管理后，下面的操作才会生效

MongoDB

```
use products
db.createUser(
   {
     user: "accountUser",
     pwd: "password",
     roles: [ "readWrite", "dbAdmin" ]
   }
)
```

```

```

早期版本

```
> use neo
switched to db neo
> db.addUser('neo','chen')
{
        "user" : "neo",
        "readOnly" : false,
        "pwd" : "68ace374737253d87e0ec91d4fcb673d"
}

> db.system.users.find()
{ "_id" : ObjectId("4c481404b9db6474d2fcb76f"), "user" : "neo",
"readOnly" : false, "pwd" : "68ace374737253d87e0ec91d4fcb673d" }

> db.auth('neo','chen')
1
```

数据库监控用户

```
db.createUser(
   {
     user: "monitor",
     pwd: "netkiller",
     roles: [ "clusterMonitor"]
   }
)
```

删除用户

Deleting Users 删除用户

To delete a user:

```
> db.getUsers();
```

```
[
        {
                "_id" : "test.monitor",
                "user" : "monitor",
                "db" : "test",
                "roles" : [
                        {
                                "role" : "dbOwner",
                                "db" : "test"
                        }
                ]
        }
]

> db.dropUser('monitor')
ture

> db.getUsers();
[ ]
```

更新角色

```
db.updateUser( "monitor",
   {
     roles: [ "read", "clusterMonitor" ]
   }
)
```

2.4.0早期旧版本

开启认证

```
# vim /etc/mongodb.conf
auth = true
```

重载配置文件

```
# /etc/init.d/mongod reload
Stopping mongod:                                           [
OK  ]
Starting mongod:                                           [
OK  ]
```

```
use admin;
db.createUser(
   {
     user: "admin",
     pwd: "WkAFdmfVQpP1oAEkz4YVlMCDxkG36TAi",
     roles: [ "dbAdmin", "dbOwner", "userAdmin" ]
   }
);
```

早期版本删除用户

```
db.system.users.remove( { user: username } )
```

第 13 章 命令工具

1. mongo - MongoDB Shell

eval

```
# mongo
MongoDB shell version: 2.2.3
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
>
```

3.4

```
[root@netkiller ~]# mongo
MongoDB shell version v3.4.1
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.1
Server has startup warnings:
2017-01-03T11:26:57.516+0800 I CONTROL  [initandlisten]
2017-01-03T11:26:57.516+0800 I CONTROL  [initandlisten] **
WARNING: Access control is not enabled for the database.
2017-01-03T11:26:57.516+0800 I CONTROL  [initandlisten] **
Read and write access to data and configuration is
unrestricted.
2017-01-03T11:26:57.516+0800 I CONTROL  [initandlisten]
2017-01-03T11:26:57.516+0800 I CONTROL  [initandlisten]
2017-01-03T11:26:57.516+0800 I CONTROL  [initandlisten] **
WARNING: /sys/kernel/mm/transparent_hugepage/enabled is
'always'.
```

```
2017-01-03T11:26:57.516+0800 I CONTROL  [initandlisten] **
We suggest setting it to 'never'
2017-01-03T11:26:57.516+0800 I CONTROL  [initandlisten]
2017-01-03T11:26:57.516+0800 I CONTROL  [initandlisten] **
WARNING: /sys/kernel/mm/transparent_hugepage/defrag is
'always'.
2017-01-03T11:26:57.516+0800 I CONTROL  [initandlisten] **
We suggest setting it to 'never'
2017-01-03T11:26:57.516+0800 I CONTROL  [initandlisten]
>
```

```
# mongo 127.0.0.1:27017/admin --eval "db.stats()"
```

help

help

```
db.help() help on DB methods
db.foo.help() help on collection methods
```

登陆认证

```
# mongo -u<user> -p<password> --authenticationDatabase <db>
<host>/<db>
```

管道操作

```
cat data.bson | mongo test
```

2. mongodump - Backup

本地备份

如果没有开启用户认证

```
root@ubuntu:~/neo# mongodump -d eos --gzip
```

开启用户认证

```
# mongodump -uneo -p -d test -o /tmp/
connected to: 127.0.0.1
Enter password:
Tue Sep 8 10:12:33.011 DATABASE: test to /tmp/test
Tue Sep 8 10:12:33.012 test.system.indexes to
/tmp/test/system.indexes.bson
Tue Sep 8 10:12:33.043 12 objects
Tue Sep 8 10:12:33.043 test.bios to /tmp/test/bios.bson
Tue Sep 8 10:12:33.043 1 objects
Tue Sep 8 10:12:33.043 Metadata for test.bios to
/tmp/test/bios.metadata.json
Tue Sep 8 10:12:33.043 test.system.users to
/tmp/test/system.users.bson
Tue Sep 8 10:12:33.044 2 objects
Tue Sep 8 10:12:33.044 Metadata for test.system.users to
/tmp/test/system.users.metadata.json
Tue Sep 8 10:12:33.044 test.fs.chunks to
/tmp/test/fs.chunks.bson
Tue Sep 8 10:12:33.045 2 objects
Tue Sep 8 10:12:33.045 Metadata for test.fs.chunks to
/tmp/test/fs.chunks.metadata.json
Tue Sep 8 10:12:33.045 test.fs.files to /tmp/test/fs.files.bson
Tue Sep 8 10:12:33.046 2 objects
Tue Sep 8 10:12:33.046 Metadata for test.fs.files to
/tmp/test/fs.files.metadata.json
```

```
Tue Sep 8 10:12:33.046 test.images.chunks to
/tmp/test/images.chunks.bson
Tue Sep 8 10:12:33.167 12 objects
Tue Sep 8 10:12:33.167 Metadata for test.images.chunks to
/tmp/test/images.chunks.metadata.json
Tue Sep 8 10:12:33.167 test.images.files to
/tmp/test/images.files.bson
Tue Sep 8 10:12:33.168 3 objects
Tue Sep 8 10:12:33.168 Metadata for test.images.files to
/tmp/test/images.files.metadata.json
Tue Sep 8 10:12:33.168 test.img.chunks to
/tmp/test/img.chunks.bson
Tue Sep 8 10:12:33.175 4 objects
Tue Sep 8 10:12:33.175 Metadata for test.img.chunks to
/tmp/test/img.chunks.metadata.json
Tue Sep 8 10:12:33.175 test.img.files to
/tmp/test/img.files.bson
Tue Sep 8 10:12:33.176 1 objects
Tue Sep 8 10:12:33.176 Metadata for test.img.files to
/tmp/test/img.files.metadata.json
```

通过指定dbpath在本地导出bson文件

```
mongodump --dbpath /var/lib/mongodb --out /opt/backup --db test
--username backup --password passwd
```

远程备份

短参数

```
mongodump -h mongodb.example.net -p 27017 -u neo -p password -d
netkiller -c yourcollection
```

长参数

```
mongodump --host mongodb.example.net --port 27017 --username
backup --password passwd --db mdb --collection some
```

3. mongorestore

本地恢复

直接从dump恢复备份

```
[root@netkiller www]# ls
backup dump

[root@netkiller www]# mongorestore dump/
```

```
mongorestore --dbpath /var/lib/mongodb --journal /opt/backu
```

远程恢复

```
[root@netkiller www]# mongorestore -h 127.0.0.1 -u neo -p chen
/tmp/test/
connected to: 127.0.0.1
Tue Sep 8 10:18:31.360 /tmp/test/system.users.bson
Tue Sep 8 10:18:31.360 going into namespace [test.system.users]
Tue Sep 8 10:18:31.361 warning: Restoring to test.system.users
without dropping. Restored data will be inserted without
raising errors; check your server log
2 objects found
Tue Sep 8 10:18:31.361 Creating index: { key: { _id: 1 }, ns:
"test.system.users", name: "_id_" }
Tue Sep 8 10:18:31.406 Creating index: { key: { user: 1,
userSource: 1 }, unique: true, ns: "test.system.users", name:
"user_1_userSource_1" }
Tue Sep 8 10:18:31.406 /tmp/test/img.chunks.bson
Tue Sep 8 10:18:31.406 going into namespace [test.img.chunks]
```

```
Tue Sep 8 10:18:31.407 warning: Restoring to test.img.chunks
without dropping. Restored data will be inserted without
raising errors; check your server log
4 objects found
Tue Sep 8 10:18:31.409 Creating index: { name: "_id_", key: {
_id: 1 }, ns: "test.img.chunks" }
Tue Sep 8 10:18:31.409 Creating index: { name:
"files_id_1_n_1", key: { files_id: 1, n: 1 }, unique: true, ns:
"test.img.chunks" }
Tue Sep 8 10:18:31.409 /tmp/test/fs.files.bson
Tue Sep 8 10:18:31.409 going into namespace [test.fs.files]
Tue Sep 8 10:18:31.410 warning: Restoring to test.fs.files
without dropping. Restored data will be inserted without
raising errors; check your server log
2 objects found
Tue Sep 8 10:18:31.410 Creating index: { name: "_id_", key: {
_id: 1 }, ns: "test.fs.files" }
Tue Sep 8 10:18:31.410 /tmp/test/images.chunks.bson
Tue Sep 8 10:18:31.410 going into namespace
[test.images.chunks]
Tue Sep 8 10:18:31.411 warning: Restoring to test.images.chunks
without dropping. Restored data will be inserted without
raising errors; check your server log
12 objects found
Tue Sep 8 10:18:31.414 Creating index: { name: "_id_", key: {
_id: 1 }, ns: "test.images.chunks" }
Tue Sep 8 10:18:31.414 Creating index: { name:
"files_id_1_n_1", key: { files_id: 1, n: 1 }, unique: true, ns:
"test.images.chunks" }
Tue Sep 8 10:18:31.414 /tmp/test/images.files.bson
Tue Sep 8 10:18:31.414 going into namespace [test.images.files]
Tue Sep 8 10:18:31.414 warning: Restoring to test.images.files
without dropping. Restored data will be inserted without
raising errors; check your server log
3 objects found
Tue Sep 8 10:18:31.415 Creating index: { name: "_id_", key: {
_id: 1 }, ns: "test.images.files" }
Tue Sep 8 10:18:31.415 /tmp/test/fs.chunks.bson
Tue Sep 8 10:18:31.415 going into namespace [test.fs.chunks]
Tue Sep 8 10:18:31.415 warning: Restoring to test.fs.chunks
without dropping. Restored data will be inserted without
raising errors; check your server log
2 objects found
Tue Sep 8 10:18:31.416 Creating index: { name: "_id_", key: {
_id: 1 }, ns: "test.fs.chunks" }
```

```
Tue Sep 8 10:18:31.416 Creating index: { name:
"files_id_1_n_1", key: { files_id: 1, n: 1 }, unique: true, ns:
"test.fs.chunks" }
Tue Sep 8 10:18:31.416 /tmp/test/img.files.bson
Tue Sep 8 10:18:31.416 going into namespace [test.img.files]
Tue Sep 8 10:18:31.417 warning: Restoring to test.img.files
without dropping. Restored data will be inserted without
raising errors; check your server log
1 objects found
Tue Sep 8 10:18:31.417 Creating index: { name: "_id_", key: {
_id: 1 }, ns: "test.img.files" }
Tue Sep 8 10:18:31.417 /tmp/test/bios.bson
Tue Sep 8 10:18:31.417 going into namespace [test.bios]
Tue Sep 8 10:18:31.417 warning: Restoring to test.bios without
dropping. Restored data will be inserted without raising
errors; check your server log
1 objects found
Tue Sep 8 10:18:31.417 Creating index: { key: { _id: 1 }, ns:
"test.bios", name: "_id_" }
```

恢复到指定数据库

```
# mongorestore -h 127.0.0.1 -d test123 /tmp/test
```

```
mongorestore --host mongodb.example.net --port 27017 --username
backup --password password --db test --collection some
/data/backup
```

filter

如果只想恢复部分数据，可以使用--filter

```
$ mongorestore -h 127.0.0.1 -d test123 /tmp/test --filter
'{"field": 1}'
```

4. mongostat

```
# mongostat
connected to: 127.0.0.1
insert query update delete getmore command flushes mapped vsize
res faults locked db idx miss % qr|qw ar|aw netIn netOut conn
time
*0 *0 *0 *0 0 1|0 0 848m 1.92g 162m 0 wechat:0.0% 0 0|0 0|0 62b
4k 1 10:38:53
*0 *0 *0 *0 0 1|0 0 848m 1.92g 162m 0 wechat:0.0% 0 0|0 0|0 62b
4k 1 10:38:54
*0 *0 *0 *0 0 1|0 0 848m 1.92g 162m 0 wechat:0.0% 0 0|0 0|0 62b
4k 1 10:38:55
*0 *0 *0 *0 0 1|0 0 848m 1.92g 162m 0 wechat:0.0% 0 0|0 0|0 62b
4k 1 10:38:56
*0 *0 *0 *0 0 1|0 0 848m 1.92g 162m 0 wechat:0.0% 0 0|0 0|0 62b
4k 1 10:38:57
```

5. mongotop

```
# mongotop
connected to: 127.0.0.1

ns total read write 2015-09-08T02:23:46
passport.system.users 0ms 0ms 0ms
passport.system.namespaces 0ms 0ms 0ms
passport.system.indexes 0ms 0ms 0ms
member.system.users 0ms 0ms 0ms
member.system.namespaces 0ms 0ms 0ms
member.system.indexes 0ms 0ms 0ms
```

6. mongofiles - Browse and modify a GridFS filesystem.

list 浏览文件

```
# mongofiles list
connected to: 127.0.0.1
/etc/passwd 2176
/tmp/test1.php 192
```

put 上传文件

```
# mongofiles put /bin/ls
connected to: 127.0.0.1
added file: { _id: ObjectId('55ee4c68bd053b7418404c53'),
filename: "/bin/ls", chunkSize: 261120, uploadDate: new
Date(1441680488106), md5: "ca226dd605e91b72e0d2060a6357c28f",
length: 109208 }
done!

# mongofiles list
connected to: 127.0.0.1
/etc/passwd 2176
/tmp/test1.php 192
/bin/ls 109208
```

上传指定数据库

```
# mongofiles put -d images -c img /etc/fstab
connected to: 127.0.0.1
added file: { _id: ObjectId('55ee4d5416377f58d0a9e714'),
filename: "/etc/fstab", chunkSize: 261120, uploadDate: new
Date(1441680724579), md5: "381185dc0c4807b88406b452b4acc2e8",
length: 1067 }
done!

# mongofiles list -d images -c img
```

```
connected to: 127.0.0.1
/etc/fstab 1067
```

collection 参数有 bug 需要注意。

-c img 似乎无效，可能是mongofiles的bug. 使用PHP测试上传是可以指定collection，并且没有任何问题。

```
# mongofiles put -d images --collection abc /etc/nfsmount.conf
connected to: 127.0.0.1
added file: { _id: ObjectId('55ee4f5ef4b26bc3189dc8a5'),
filename: "/etc/nfsmount.conf", chunkSize: 261120, uploadDate:
new Date(1441681246083), md5:
"ce3b9fee8612087cbb69d46db34ce8ec", length: 3605 }
done!

# mongofiles -d images --collection abc list
connected to: 127.0.0.1
/etc/fstab      1067
/etc/passwd     2555
/etc/goaccess.conf      6956
/etc/krb5.conf  449
/etc/nfsmount.conf      3605

# mongo images
> show collections;
abc.fs.chunks
abc.fs.files
fs.chunks
fs.files
system.indexes
>

> db.abc.fs.files.find();
>
```

使用 --collection 参数可以看到abc已经创建，但我们去db.abc.fs.files.find();发现里面没有任何数据，文件仍然被上传到abc.fs.files

get 下载

如果 /tmp/test123 存在则会覆盖

```
# mongofiles get /tmp/test123
connected to: 127.0.0.1
done write to: /tmp/test123
```

-l 指定路径，相当于另存。

```
# mongofiles get /tmp/test123 -l /tmp/aabbcc
connected to: 127.0.0.1
done write to: /tmp/aabbcc
```

delete 删除

```
# mongofiles list
connected to: 127.0.0.1
/etc/passwd 2176
/tmp/test1.php 192
/bin/ls 109208
/tmp/test123 6

# mongofiles delete /tmp/test123
connected to: 127.0.0.1
done!

# mongofiles list
connected to: 127.0.0.1
/etc/passwd 2176
/tmp/test1.php 192
/bin/ls 109208
```

第 14 章 MongoDB Shell

1. shutdownServer

关闭MongoDB数据库

```
db.shutdownServer()
```

2. show 查看命令

show dbs

show dbs show database names

```
> show dbs
local   (empty)
logging 0.203125GB
test    0.203125GB
```

show collections

show collections show collections in current database

```
> show collections
bios
system.indexes
```

另一种用法是show tables

```
> show tables
bios
system.indexes
```

show users

show users show users in current database

```

```

show profile

show profile show most recent system.profile entries with time >= 1ms

```
> show profile
db.system.profile is empty
Use db.setProfilingLevel(2) will enable profiling
Use db.system.profile.find() to show raw profile entries
```

3. 切换数据库

```
use <db name>                    set curent database to <db name>

> use logging
switched to db logging
```

4. save

存储嵌套的对象

```
db.foo.save({'name':'neo','address':
{'city':'shenzhen','post':518000},'phone':
[13113668890,13322993040]})
```

存储数组对象

```
db.foo.save({'Uid':'netkiller@msn.com','phone':
['13322993040','13113668890']})
```

5. insert

```
db.bios.insert(
   {
     _id: 1,
     name: { first: 'John', last: 'Backus' },
     birth: new Date('Dec 03, 1924'),
     death: new Date('Mar 17, 2007'),
     contribs: [ 'Fortran', 'ALGOL', 'Backus-Naur Form', 'FP'
],
     awards: [
                {
                  award: 'W.W. McDowell Award',
                  year: 1967,
                  by: 'IEEE Computer Society'
                },
                {
                  award: 'National Medal of Science',
                  year: 1975,
                  by: 'National Science Foundation'
                },
                {
                  award: 'Turing Award',
                  year: 1977,
                  by: 'ACM'
                },
                {
                  award: 'Draper Prize',
                  year: 1993,
                  by: 'National Academy of Engineering'
                }
              ]
   }
)
```

6. update

根据query条件修改，如果不存在则插入，允许修改多条记录

```
                        db.foo.update({'yy':5},{'$set':
{'xx':2}},upsert=true,multi=true)
```

multi 更新所有数据

update 第一个参数是条件，当不写条件时将匹配所有数据。

```
db.getCollection('certificate').update({},{'$set':
{'icon':'52bfbb7d92b3f41da2e4103f1990c054990be863.png'}},upsert
=false,multi=true)
```

upsert 更新，如果不存在则插入数据

```
db.getCollection('shippingAddress').update({'memberId':'0000000
0'},{'$set':{'defaults': false}},upsert=true,multi=true)
```

7. remove

删除uid=10的记录

```
db.foo.remove({'uid':10})
```

删除所有的记录

```
db.foo.remove()
```

删除条件使用 _id

```
db.foo.remove({ "_id" :
ObjectId("56e10b66a22ef1b1408b4567")})

db.getCollection('goods').remove({ "_id":
ObjectId("5bbdbd197099aa06abf6fb1a")})
```

8. 删除 collection

```
db.collection.drop()
```

删除字段

```
db.getCollection('table').update({},{$unset:{field:1}})
```

9. count()

```
> db.access.count()
51528
> db.access.count()
104401
```

10. 查询

find() MongoDB 2.x

查找所有 所有记录

```
                    db.foo.find() list objects in collection foo
                    db.foo.find( { a : 1 } ) list objects in foo
where a == 1
```

查找一条记录

```
                    db.foo.findOne()
```

根据条件检索10条记录

```
                    db.foo.find({'name':'neo'}).limit(10)
```

sort排序

```
                    db.foo.find({'name':'neo'}).sort({'Dt',-1})
                    db.foo.find().sort({'Ct':-1}).limit(1)
```

count记录统计操作

```
                    db.foo.count()
```

distinct操作,去重复查询指定列,

```
                    db.foo.distinct('name')
```

”>=”操作

```
db.foo.find({"timestamp": {"$gte" : 2}})
```

子对象的查找

```
                        db.foo.find({'address.city':'shenzhen'})
```

find() MongoDB 3.x

```
db.getCollection('tracker').find({name:"81004892"})
```

Query

包含字段

```
db.getCollection('pyramidSelling').find({},{'phone':1})
```

排除字段

```
db.getCollection('pyramidSelling').find({},{'phone':0})
```

sort()

```
db.getCollection('tracker').find({name:"81004892"}).sort({ctime: -1})
```

group()

group()类似SQL中的Group by

```
> db.test.group({key: {remote_addr: true}, initial: {count: 0}, reduce:
function(obj, prev) {prev.count++}});
[
        {
                "remote_addr" : "192.168.2.76",
                "count" : 3
        },
        {
                "remote_addr" : "192.168.2.70",
                "count" : 1
        }
]
```

11. aggregate

project

$split

```
                                        {
                                        "_id" :
ObjectId("591a710320156761bdf68a06"),
                                        "_class" :
"mis.domain.PyramidSelling",

                                        ...
                                        ...

                                        "status" : true,
                                        "createdDate" :
ISODate("2017-05-16T03:24:51.511Z")
                                        }
```

```
db.getCollection('pyramidSelling').aggregate([
  { $project : { _class : { $split: ["$_class", "."] } } }
]);
```

substr

```
db.getCollection('pyramidSelling').aggregate(
                                        [
                                        {
                                        $project: {
                                        userName: 1,
                                        phone: {
                                        prefix: { $substr: [
```

```
"$phone", 0, 3 ] },
                                                    mobile: { $substr: [
"$phone", 3, 11 ] }
                                                    },
                                                    }
                                                    }
                                                    ]
                                                    )
```

groupby + sum

select username, sum(balance) as total from users group by member.

```
db.member.aggregate([{
    $group: {
        _id: "$username",
        total: { $sum: "$balance" }
    }
}])
```

12. Indexes 索引

增加索引： 1(ascending),-1(descending)

查看索引

```
db.getCollection('product').getIndexes()
```

```
[
    {
        "v" : 2,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_",
        "ns" : "netkiller.product"
    },
    {
        "v" : 2,
        "unique" : true,
        "key" : {
            "uuid" : 1
        },
        "name" : "uuid",
        "ns" : "netkiller.product",
        "sparse" : true
    },
    {
        "v" : 2,
        "key" : {
            "nfc" : 1
        },
        "name" : "nfc",
        "ns" : "netkiller.product"
    },
    {
        "v" : 2,
        "unique" : true,
        "key" : {
            "qrcode" : 1
        },
        "name" : "qrcode",
```

```
        "ns" : "netkiller.product",
        "sparse" : true
    },
    {
        "v" : 2,
        "key" : {
            "memberId" : 1
        },
        "name" : "memberId",
        "ns" : "netkiller.product"
    },
    {
        "v" : 2,
        "unique" : true,
        "key" : {
            "transactionId" : 1
        },
        "name" : "transactionId",
        "ns" : "netkiller.product",
        "sparse" : true
    }
]
```

查看索引信息

```
db.logging.getIndexes()
[
        {
                "v" : 1,
                "key" : {
                        "_id" : 1
                },
                "ns" : "logging.logging",
                "name" : "_id_"
        }
]
```

查看索引名与排序方式

```
db.getCollection('member').getIndexKeys();

[
```

```
    {
        "_id" : 1
    },
    {
        "mobile" : 1
    },
    {
        "username" : 1
    },
    {
        "wechat" : 1
    }
]
```

创建索引

增加索引

```
db.foo.ensureIndex({firstname: 1, lastname: 1}, {unique: true});
```

索引子对象

```
db.logging.users.ensureIndex({address.city:1})
```

删除索引

```
db.getCollection('product').dropIndex("memberId")
```

根据索引名删除索引

```
> db.logging.users.dropIndex('name_1')
```

```
{ "nIndexesWas" : 2, "ok" : 1 }

> db.logging.users.getIndexKeys()
[ { "_id" : 1 } ]
```

唯一索引

```
db.members.createIndex( { "user_id": 1 }, { unique: true } )
```

```
> db.apple.createIndex({"devicetoken":1},{unique: true})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

复合索引

```
db.getCollection('foo').ensureIndex({"address":1,"phone":1})
```

稀疏索引

```
db.getCollection('article').ensureIndex({"uuid": 1}, {"unique":
true,"sparse":true});
```

作用,唯一索引只允许一条索引字段为空的记录存在，之后就不允许插入了。再次插入为 null 的记录时会报错:

```
E11000 duplicate key error index: dup key: { : null };
```

“sparse”的作用就是当 uuid 在文档中不存在，或为空值，则不进入索引，从而避免上述问题。

13. Map-Reduce

使用 Map-Reduce 统计Web 服务器 access.log日 志文件

首先将web服务器access.log导入到mongodb,参考 http://netkiller.github.io/article/log.html 格式如下:

```
{
        "_id" : ObjectId("51553efcd8616be7e5395c0d"),
        "remote_addr" : "192.168.2.76",
        "remote_user" : "-",
        "time_local" : "29/Mar/2013:09:20:31 +0800",
        "request" : "GET /tw/ad.jpg HTTP/1.1",
        "status" : "200",
        "body_bytes_sent" : "5557",
        "http_referer" : "http://www.example.com/tw/",
        "http_user_agent" : "Mozilla/5.0 (Windows NT 6.1;
WOW64) AppleWebKit/537.17 (KHTML, like Gecko)
Chrome/24.0.1312.57 Safari/537.17",
        "http_x_forwarded_for" : "-"
}
```

创建map方法

```
var mapFunction1 = function() {
    emit(this.remote_addr, {count:1});
};
```

创建reduce方法

```
var reduceFunction1 = function(key, values) {
        var total = 0;
        values.forEach(function (value) {total +=
value.count;});
    return {ipaddr: key, count:total};
```

```
};
```

分析数据

```
db.access.mapReduce(mapFunction1, reduceFunction1, {out :
"resultCollection"});
```

输出结果

```
db.resultCollection.find();
```

Demo 数据库

```
> db.resultCollection.find();
{ "_id" : "192.168.2.109", "value" : { "ipaddr" :
"192.168.2.109", "count" : 554 } }
{ "_id" : "192.168.2.38", "value" : { "ipaddr" :
"192.168.2.38", "count" : 26 } }
{ "_id" : "192.168.2.39", "value" : { "ipaddr" :
"192.168.2.39", "count" : 72 } }
{ "_id" : "192.168.2.40", "value" : { "ipaddr" :
"192.168.2.40", "count" : 3564 } }
{ "_id" : "192.168.2.49", "value" : { "ipaddr" :
"192.168.2.49", "count" : 955 } }
{ "_id" : "192.168.2.5", "value" : { "ipaddr" : "192.168.2.5",
"count" : 2 } }
{ "_id" : "192.168.2.76", "value" : { "ipaddr" :
"192.168.2.76", "count" : 60537 } }
{ "_id" : "192.168.3.12", "value" : { "ipaddr" :
"192.168.3.12", "count" : 9577 } }
{ "_id" : "192.168.3.14", "value" : { "ipaddr" :
"192.168.3.14", "count" : 343 } }
{ "_id" : "192.168.3.18", "value" : { "ipaddr" :
"192.168.3.18", "count" : 1006 } }
{ "_id" : "192.168.3.26", "value" : { "ipaddr" :
"192.168.3.26", "count" : 2714 } }
```

```
{ "_id" : "192.168.6.19", "value" : { "ipaddr" :
"192.168.6.19", "count" : 668 } }
{ "_id" : "192.168.6.2", "value" : { "ipaddr" : "192.168.6.2",
"count" : 123760 } }
{ "_id" : "192.168.6.30", "value" : { "ipaddr" :
"192.168.6.30", "count" : 1196 } }
{ "_id" : "192.168.6.35", "value" : { "ipaddr" :
"192.168.6.35", "count" : 1050 } }
>
```

14.内嵌对象

Array / List 列表类型

```
db.foo.save(
{
        'name':'neo',
        'address':[{'city':'shenzhen','post':"518000"},
{'city':'heilongjiang','post':"135000"}],
        'phone':["13113668800","13322993040","13266884444"]
}
)
```

```
{
    "_id" : ObjectId("5bbefed1b04a8c0d7395d1b5"),
    "name" : "neo",
    "address" : [
        {
            "city" : "shenzhen",
            "post" : "518000"
        },
        {
            "city" : "heilongjiang",
            "post" : "135000"
        }
    ],
    "phone" : [
        "13113668800",
        "13322993040",
        "13266884444"
    ]
}
```

删除数组元素

```
db.getCollection('foo').update({"_id":ObjectId("5bbeff40b04a8c0
d7395d1b6")},{"$pull":{"phone":"13266884444"}})
```

删除数组中的对象

```
db.getCollection('foo').update({"_id":ObjectId("5bbeff40b04a8c0
d7395d1b6")},{"$pull":{"address":{"city":"heilongjiang"}}})
```

查找替换

```
db.getCollection('foo').update({"address.city":"shenzhen"},
{"$set":{"address.$.post":"000000"}})
```

15. Javascript 脚本

```
db.numbers.drop()

var counter = 0
while (counter<=1000){
  db.numbers.save({"value":counter})
  counter = counter + 1;
}
```

第 15 章 Mongo Admin UI

http://docs.mongodb.org/ecosystem/tools/administration-interfaces/

1. RockMongo

http://code.google.com/p/rock-php/

2. MongoVUE

http://blog.mongovue.com/

MongoVUE 是收费软件，提供的功能比较完善，可以免费试用

第 16 章 Cassandra

http://incubator.apache.org/cassandra/

1. Getting Started

1.1. Downloading and Installation

$ cd /srv/

```
$ cd /usr/local/src/

$ sudo wget -c
http://apache.freelamp.com/cassandra/0.5.1/apache-cassandra-
0.5.1-bin.tar.gz
$ sudo tar zxvf apache-cassandra-0.5.1-bin.tar.gz
$ cp -r /usr/local/src/apache-cassandra-0.5.1 /srv/
$ cd /srv/
$ sudo ln -s apache-cassandra-0.5.1 apache-cassandra
$ cd apache-cassandra
```

1.2. Running Cassandra

Running Cassandra

```
$ bin/cassandra
$ Listening for transport dt_socket at address: 8888
INFO - Saved Token not found. Using
70882909557229809272696372631016976044
INFO - Starting up server gossip
```

1.3. cli tool

cli

$ bin/cassandra-cli

```
neo@db:/srv/apache-cassandra$ bin/cassandra-cli
Welcome to cassandra CLI.

Type 'help' or '?' for help. Type 'quit' or 'exit' to quit.
cassandra>
```

```
cassandra> connect localhost/9160
Connected to localhost/9160
```

1.4. Testing Cassandra

test

```
cassandra> show keyspaces
Keyspace1
system
```

insert value

```
cassandra> set Keyspace1.Standard1['member']['name']='neo'
Value inserted.
cassandra> set Keyspace1.Standard1['member']['age']='27'
```

```
Value inserted.
cassandra> set Keyspace1.Standard1['member']
['email']='openunix@163.com'
Value inserted.
cassandra>
cassandra> get Keyspace1.Standard1['member']
=> (column=name, value=neo, timestamp=1271070497471)
=> (column=email, value=openunix@163.com,
timestamp=1271070498334)
=> (column=age, value=27, timestamp=1271070497519)
Returned 3 results.
cassandra>
```

2. Configure Cassandra

2.1. Envionment variables

```
CASSANDRA_HOME=/srv/apache-cassandra
```

2.2. log4j.properties

```
[root@db apache-cassandra]# vim conf/log4j.properties
log4j.appender.R=org.apache.log4j.RollingFileAppender
log4j.appender.file.maxFileSize=20MB
log4j.appender.file.maxBackupIndex=50
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%5p [%t] %d{ISO8601}
%F (line %L) %m%n
# Edit the next line to point to your logs directory
log4j.appender.R.File=/var/log/cassandra/system.log

# Application logging options
#log4j.logger.com.facebook=DEBUG
#log4j.logger.com.facebook.infrastructure.gms=DEBUG
#log4j.logger.com.facebook.infrastructure.db=DEBUG
```

2.3. storage-conf.xml

```
[root@db apache-cassandra]# vim conf/storage-conf.xml
```

3. Keyspace

3.1. Schema

Keyspace

Column family

Name

Column

Super column

Sorting

3.2. Keyspace example

例 16.1. Twitter

```
<Keyspace Name="Twitter">
<ColumnFamily CompareWith="UTF8Type" Name="Statuses" />
<ColumnFamily CompareWith="UTF8Type" Name="StatusAudits" />
<ColumnFamily CompareWith="UTF8Type" Name="StatusRelationships"
CompareSubcolumnsWith="TimeUUIDType" ColumnType="Super" />
<ColumnFamily CompareWith="UTF8Type" Name="Users" />
<ColumnFamily CompareWith="UTF8Type" Name="UserRelationships"
CompareSubcolumnsWith="TimeUUIDType" ColumnType="Super" />
</Keyspace>
```

例 16.2. Twissandra

```
  <Keyspaces>
```

```
    <Keyspace Name="Twissandra">
       <ColumnFamily CompareWith="UTF8Type" Name="User"/>
      <ColumnFamily CompareWith="BytesType" Name="Username"/>
      <ColumnFamily CompareWith="BytesType" Name="Friends"/>
      <ColumnFamily CompareWith="BytesType" Name="Followers"/>
      <ColumnFamily CompareWith="UTF8Type" Name="Tweet"/>
      <ColumnFamily CompareWith="LongType" Name="Timeline"/>
      <ColumnFamily CompareWith="LongType" Name="Userline"/>


<ReplicaPlacementStrategy>org.apache.cassandra.locator.RackUnaw
areStrategy</ReplicaPlacementStrategy>

      <!-- Number of replicas of the data -->
      <ReplicationFactor>1</ReplicationFactor>

<EndPointSnitch>org.apache.cassandra.locator.EndPointSnitch</En
dPointSnitch>

    </Keyspace>
  </Keyspaces>
```

Schema Layout

In Cassandra, the way that your data is structured is very closely tied to how how it will be retrieved. Let's start with the user ColumnFamily. The key is a user id, and the columns are the properties on the user:

```
User = {
    'a4a70900-24e1-11df-8924-001ff3591711': {
        'id': 'a4a70900-24e1-11df-8924-001ff3591711',
        'username': 'ericflo',
        'password': '****',
    },
}
```

Since some of the URLs on the site actually have the username, we need to be able to map from the username to the user id:

```
Username = {
    'ericflo': {
        'id': 'a4a70900-24e1-11df-8924-001ff3591711',
    },
}
```

Friends and followers are keyed by the user id, and then the columns are the friend user id and follower user ids, and we store a timestamp as the value because it's interesting information to have:

```
Friends = {
    'a4a70900-24e1-11df-8924-001ff3591711': {
        # friend id: timestamp of when the friendship was added
        '10cf667c-24e2-11df-8924-001ff3591711':
'1267413962580791',
        '343d5db2-24e2-11df-8924-001ff3591711':
'1267413990076949',
        '3f22b5f6-24e2-11df-8924-001ff3591711':
'1267414008133277',
    },
}

Followers = {
    'a4a70900-24e1-11df-8924-001ff3591711': {
        # friend id: timestamp of when the followership was
added
        '10cf667c-24e2-11df-8924-001ff3591711':
'1267413962580791',
        '343d5db2-24e2-11df-8924-001ff3591711':
'1267413990076949',
        '3f22b5f6-24e2-11df-8924-001ff3591711':
'1267414008133277',
    },
}
```

Tweets are stored in a way similar to users:

```
Tweet = {
```

```
    '7561a442-24e2-11df-8924-001ff3591711': {
        'id': '89da3178-24e2-11df-8924-001ff3591711',
        'user_id': 'a4a70900-24e1-11df-8924-001ff3591711',
        'body': 'Trying out Twissandra. This is awesome!',
        '_ts': '1267414173047880',
    },
}
```

The Timeline and Userline column families keep track of which tweets should appear, and in what order. To that effect, the key is the user id, the column name is a timestamp, and the column value is the tweet id:

```
Timeline = {
    'a4a70900-24e1-11df-8924-001ff3591711': {
        # timestamp of tweet: tweet id
        1267414247561777: '7561a442-24e2-11df-8924-
001ff3591711',
        1267414277402340: 'f0c8d718-24e2-11df-8924-
001ff3591711',
        1267414305866969: 'f9e6d804-24e2-11df-8924-
001ff3591711',
        1267414319522925: '02ccb5ec-24e3-11df-8924-
001ff3591711',
    },
}

Userline = {
    'a4a70900-24e1-11df-8924-001ff3591711': {
        # timestamp of tweet: tweet id
        1267414247561777: '7561a442-24e2-11df-8924-
001ff3591711',
        1267414277402340: 'f0c8d718-24e2-11df-8924-
001ff3591711',
        1267414305866969: 'f9e6d804-24e2-11df-8924-
001ff3591711',
        1267414319522925: '02ccb5ec-24e3-11df-8924-
001ff3591711',
    },
}
```

4. Cluster

4.1. Running a cluster

```
<Seed>127.0.0.1</Seed>
```

改为

```
<Seed>172.16.0.1</Seed>
```

```

```

```
  <ListenAddress>localhost</ListenAddress>
改为:
  <ListenAddress>172.16.0.1</ListenAddress>
```

```
    <ThriftAddress>localhost</ThriftAddress>
改为:
    <ThriftAddress>0.0.0.0</ThriftAddress>
```

$ bin/cassandra

4.2. Running a single node

```
<Seed>127.0.0.1</Seed>
```

改为

```
<Seed>172.16.0.2</Seed>
```

```
  <Seeds>
      <Seed>172.16.0.1</Seed>
      <Seed>172.16.0.2</Seed>
      <Seed>172.16.0.3</Seed>
      <Seed>172.16.0.4</Seed>
      <Seed>172.16.0.5</Seed>
  </Seeds>
```

```
  <ListenAddress>localhost</ListenAddress>
改为:
  <ListenAddress>172.16.0.2</ListenAddress>
```

```
    <ThriftAddress>localhost</ThriftAddress>
改为:
    <ThriftAddress>0.0.0.0</ThriftAddress>
```

$ bin/cassandra

4.3. nodetool

```
nodeprobe -host 172.16.0.1 ring
```

第 17 章 Hypertable

http://hypertable.org/

1. Hypertable 安装

Hypertable 的几种安装方式

单机：安装于单机，采用本地文件系统

Hadoop：分布式安装，使用Hadoop(HDFS)作为存储

MapR：分布式安装，在MapR之上

Ceph：分布式安装，在Ceph之上

1.1. Hypertable standalone 单机安装

过程 17.1. Hypertable standalone 安装过程

1. 安装 Hypertable 软件包

```
# cd /usr/local/src/
# wget
http://cdn.hypertable.com/packages/0.9.7.0/hypertable-
0.9.7.0-linux-x86_64.rpm
```

2. 安装 Hypertable， 我个人比较喜欢用yum localinstall他会解决软件之间的依赖关系

```
# yum localinstall hypertable-0.9.7.0-linux-x86_64.rpm
```

相关的软件会自动安装

```
Dependencies Resolved

================================================================================
 Package                         Arch          Version
Repository                                  Size
================================================================================
Installing:
 hypertable                      x86_64        0.9.7.0-1
/hypertable-0.9.7.0-linux-x86_64        288 M
Installing for dependencies:
 mailcap                         noarch        2.1.31-2.el6
base                                         27 k
 perl-Bit-Vector                 x86_64        7.1-2.el6
base                                        169 k
 perl-Carp-Clan                  noarch        6.03-2.el6
base                                         25 k
 perl-Compress-Raw-Zlib          x86_64        1:2.020-
127.el6          base
68 k
 perl-Compress-Zlib              x86_64        2.020-127.el6
base                                         43 k
 perl-HTML-Parser                x86_64        3.64-2.el6
base                                        109 k
 perl-HTML-Tagset                noarch        3.20-4.el6
base                                         17 k
 perl-IO-Compress-Base           x86_64        2.020-127.el6
base                                         67 k
 perl-IO-Compress-Zlib           x86_64        2.020-127.el6
base                                        134 k
 perl-IO-String                  noarch        1.08-9.el6
base                                         15 k
 perl-URI                        noarch        1.40-2.el6
base                                        117 k
 perl-libwww-perl                noarch        5.833-2.el6
base                                        387 k

Transaction Summary
================================================================================
```

```
================================================================
Install      13 Package(s)
```

3. Hypertable 默认安装在 /opt/hypertable/0.9.7.0

   备份配置文件,

```
# cd /opt/hypertable/0.9.7.0/conf
# cp hypertable.cfg hypertable.cfg.original
```

4. FHS-IZE 安装

```
# bin/fhsize.sh
Setting up /var/opt/hypertable
Setting up /etc/opt/hypertable
fshize /opt/hypertable/0.9.7.0:  success
```

5. 设计 "CURRENT" 连接

```
# cd /opt/hypertable
# ln -s 0.9.7.0 current
```

6. 安装 notification-hook.sh 脚本.

```
# cp conf/notification-hook.tmpl conf/notification-hook.sh
# chmod o+x conf/notification-hook.sh
```

   测试 notification-hook.sh脚本 .

```
/opt/hypertable/current/conf/notification-hook.sh "Test
Message" "This is a test."
```

7. 启动 hypertable

```
# /opt/hypertable/current/bin/start-all-servers.sh local
DFS broker: available file descriptors: 1024
Started DFS Broker (local)
Started Hyperspace
Started Hypertable.Master
/proc/sys/vm/swappiness = 60
Started Hypertable.RangeServer
Started ThriftBroker
```

```
# /opt/hypertable/current/bin/ht shell

Welcome to the hypertable command interpreter.
For information about Hypertable, visit
http://hypertable.com

Type 'help' for a list of commands, or 'help shell' for a
list of shell meta commands.

hypertable>
```

8. 测试安装是否有效

```
# /opt/hypertable/current/bin/ht shell

Welcome to the hypertable command interpreter.
For information about Hypertable, visit
http://hypertable.com

Type 'help' for a list of commands, or 'help shell' for a
list of shell meta commands.

hypertable> help

USE ............... Sets the current namespace
```

```
CREATE NAMESPACE ... Creates a new namespace
DROP NAMESPACE ..... Removes a namespace
EXISTS TABLE ....... Check if table exists
CREATE TABLE ....... Creates a table
DELETE ............. Deletes all or part of a row from a
table
DESCRIBE TABLE ..... Displays a table's schema
DROP TABLE ......... Removes a table
RENAME TABLE ....... Renames a table
DUMP TABLE ......... Create efficient backup file
ALTER TABLE ........ Add/remove column family from existing
table
INSERT ............. Inserts data into a table
LOAD DATA INFILE ... Loads data from a TSV input file into
a table
SELECT ............. Selects (and display) cells from a
table
SHOW CREATE TABLE .. Displays CREATE TABLE command used to
create table
SHOW TABLES ........ Displays only the list of tables in
the current namespace
GET LISTING ........ Displays the list of tables and
namespace in the current namespace

Statements must be terminated with ';'.  For more
information on
a specific statement, type 'help <statement>', where
<statement> is from
the preceding list.

hypertable>quit
```

9. 停止 hypertable

   运行下列命令停止 Hypertable

```
$ /opt/hypertable/current/bin/stop-servers.sh
```

1.2. Hypertable on HDFS(hadoop) 安装

Hadoop - HDFS 安装指南

过程 17.2. Hypertable on HDFS

1. 创建工作目录

```
$ hadoop fs -mkdir /hypertable
$ hadoop fs -chmod 777 /hypertable
```

2. 安装 Java 运行环境

```
yum install java-1.7.0-openjdk
yum localinstall
http://ftp.cuhk.edu.hk/pub/packages/apache.org/hadoop/commo
n/hadoop-1.1.2/hadoop-1.1.2-1.x86_64.rpm
```

3. 修改 jrun bug

```
cp /opt/hypertable/current/bin/jrun
/opt/hypertable/current/bin/jrun.old

vim /opt/hypertable/current/bin/jrun
#HT_JAR=`ls -1 /opt/hypertable/doug/current/lib/java/*.jar
| grep "hypertable-[^-]*.jar" | awk 'BEGIN {FS="/"} {print
$NF}'`
HT_JAR=`ls -1 /opt/hypertable/current/lib/java/*.jar | grep
"hypertable-[^-]*.jar" | awk 'BEGIN {FS="/"} {print $NF}'`
```

```
 export JAVA_HOME=/usr
 export HADOOP_HOME=/usr
 export HYPERTABLE_HOME=/opt/hypertable/current
```

4. hypertable.cfg

```
# cat conf/hypertable.cfg
#
# hypertable.cfg
#

# HDFS Broker
#HdfsBroker.Hadoop.ConfDir=/etc/hadoop/conf
HdfsBroker.Hadoop.ConfDir=/etc/hadoop

# Ceph Broker
CephBroker.MonAddr=192.168.6.2:6789

# Local Broker
DfsBroker.Local.Root=fs/local

# DFS Broker - for clients
DfsBroker.Port=38030

# Hyperspace
Hyperspace.Replica.Host=localhost
Hyperspace.Replica.Port=38040
Hyperspace.Replica.Dir=hyperspace

# Hypertable.Master
#Hypertable.Master.Host=localhost
Hypertable.Master.Port=38050

# Hypertable.RangeServer
Hypertable.RangeServer.Port=38060

Hyperspace.KeepAlive.Interval=30000
Hyperspace.Lease.Interval=1000000
Hyperspace.GracePeriod=200000

# ThriftBroker
ThriftBroker.Port=38080
```

Hadoop 配置文件 /etc/hadoop/core-site.xml

```
# cat /etc/hadoop/core-site.xml
```

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
    <property>
         <name>fs.default.name</name>
         <value>hdfs://namenode.example.com:9000</value>
    </property>
    <property>
         <name>hadoop.tmp.dir</name>
         <value>/var/tmp/hadoop</value>
    </property>
</configuration>
```

Hadoop 配置文件 /etc/hadoop/hdfs-site.xml

```
# cat /etc/hadoop/hdfs-site.xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
    <property>
        <name>dfs.name.dir</name>
        <value>/var/hadoop/name1</value>
        <description>  </description>
    </property>
    <property>
        <name>dfs.data.dir</name>
        <value>/var/hadoop/hdfs/data1</value>
        <description> </description>
    </property>
    <property>
        <name>dfs.replication</name>
        <value>2</value>
    </property>
</configuration>
```

```
```

5. 启动 dfsbroker

```
# /opt/hypertable/current/bin/set-hadoop-distro.sh cdh4
Hypertable successfully configured for Hadoop cdh4
```

```
# /opt/hypertable/current/bin/start-dfsbroker.sh hadoop
DFS broker: available file descriptors: 1024
Started DFS Broker (hadoop)
```

查看启动日志

```
# tail -f /opt/hypertable/current/log/DfsBroker.hadoop.log
log4j:WARN No appenders could be found for logger
(org.apache.hadoop.conf.Configuration).
log4j:WARN Please initialize the log4j system properly.
HdfsBroker.dfs.client.read.shortcircuit=false
HdfsBroker.dfs.replication=2
HdfsBroker.Server.fs.default.name=hdfs://namenode.example.c
om:9000
Apr 23, 2013 6:43:18 PM org.hypertable.AsyncComm.IOHandler
DeliverEvent
INFO: [/192.168.6.25:53556 ; Tue Apr 23 18:43:18 HKT 2013]
Connection Established
Apr 23, 2013 6:43:18 PM
org.hypertable.DfsBroker.hadoop.ConnectionHandler handle
INFO: [/192.168.6.25:53556 ; Tue Apr 23 18:43:18 HKT 2013]
Disconnect - COMM broken connection : Closing all open
handles from /192.168.6.25:53556
Closed 0 input streams and 0 output streams for client
connection /192.168.6.25:53556
```

1.3. MapR

1.4. Ceph

修改 CephBroker.MonAddr 对应的服务器与端口 号即 可

```
# cat hypertable.cfg
#
# hypertable.cfg
#

# HDFS Broker
HdfsBroker.Hadoop.ConfDir=/etc/hadoop/conf

# Ceph Broker
CephBroker.MonAddr=192.168.6.2:6789

# Local Broker
DfsBroker.Local.Root=fs/local

# DFS Broker - for clients
DfsBroker.Port=38030

# Hyperspace
Hyperspace.Replica.Host=localhost
Hyperspace.Replica.Port=38040
Hyperspace.Replica.Dir=hyperspace

# Hypertable.Master
Hypertable.Master.Port=38050

# Hypertable.RangeServer
Hypertable.RangeServer.Port=38060

Hyperspace.KeepAlive.Interval=30000
Hyperspace.Lease.Interval=1000000
Hyperspace.GracePeriod=200000

# ThriftBroker
ThriftBroker.Port=38080
```

启动 dfsbroker

```
# /opt/hypertable/current/bin/start-dfsbroker.sh ceph
```

1.5. 检验安装

创建一个表

```
# echo "USE '/'; CREATE TABLE foo ( c1, c2 ); GET LISTING;" \
>     | /opt/hypertable/current/bin/ht shell --batch
foo
sys     (namespace)
tmp     (namespace)
```

插入一些数据

```
# echo "USE '/'; INSERT INTO foo VALUES('001', 'c1', 'very'), \
>     ('000', 'c1', 'Hypertable'), ('001', 'c2', 'easy'),
('000', 'c2', 'is');" \
>     | /opt/hypertable/current/bin/ht shell --batch
```

查询数据

```
# echo "USE '/'; SELECT * FROM foo;" \
>     | /opt/hypertable/current/bin/ht shell --batch
000     c1      Hypertable
000     c2      is
001     c1      very
001     c2      easy
```

如果你想清楚所有表运行下面命令

```
$ /opt/hypertable/current/bin/stop-servers.sh
```

```
$ /opt/hypertable/current/bin/clean-database.sh
```

2. Code examples

http://hypertable.com/documentation/code_examples/

2.1. PHP

设置环境变量

```
export PHPTHRIFT_ROOT=/opt/hypertable/current/lib/php
```

安装PHP环境

```
# yum install php-cli
```

建立测试文件

```
# vim lib/php/test.php
<?php
if (!isset($GLOBALS['THRIFT_ROOT']))
    $GLOBALS['THRIFT_ROOT'] = getenv('PHPTHRIFT_ROOT');

require_once dirname(__FILE__).'/ThriftClient.php';

$client = new Hypertable_ThriftClient("localhost", 38080);
$namespace = $client->namespace_open("");

echo "HQL examples\n";
print_r($client->hql_query($namespace, "show tables"));
print_r($client->hql_query($namespace, "select * from foo"));
```

运行测试程序

```
# php lib/php/test.php
HQL examples
Hypertable_ThriftGen_HqlResult Object
(
    [results] => Array
        (
            [0] => foo
        )

    [cells] =>
    [scanner] =>
    [mutator] =>
)
Hypertable_ThriftGen_HqlResult Object
(
    [results] =>
    [cells] => Array
        (
            [0] => Hypertable_ThriftGen_Cell Object
                (
                    [key] => Hypertable_ThriftGen_Key Object
                        (
                            [row] => 000
                            [column_family] => c1
                            [column_qualifier] =>
                            [timestamp] => 1361518099519878001
                            [revision] => 1361518099519878001
                            [flag] => 255
                        )

                    [value] => Hypertable
                )

            [1] => Hypertable_ThriftGen_Cell Object
                (
                    [key] => Hypertable_ThriftGen_Key Object
                        (
                            [row] => 000
                            [column_family] => c2
                            [column_qualifier] =>
                            [timestamp] => 1361518099519878002
                            [revision] => 1361518099519878002
                            [flag] => 255
```

```
                        )

                    [value] => is
                )

            [2] => Hypertable_ThriftGen_Cell Object
                (
                    [key] => Hypertable_ThriftGen_Key Object
                        (
                            [row] => 001
                            [column_family] => c1
                            [column_qualifier] =>
                            [timestamp] => 1361518099519878003
                            [revision] => 1361518099519878003
                            [flag] => 255
                        )

                    [value] => very
                )

            [3] => Hypertable_ThriftGen_Cell Object
                (
                    [key] => Hypertable_ThriftGen_Key Object
                        (
                            [row] => 001
                            [column_family] => c2
                            [column_qualifier] =>
                            [timestamp] => 1361518099519878004
                            [revision] => 1361518099519878004
                            [flag] => 255
                        )

                    [value] => easy
                )

        )

    [scanner] =>
    [mutator] =>
)
```

3. HQL

3.1. namespace 命名空间管理

```
hypertable> create namespace test;
  Elapsed time:  0.22 s
hypertable> use test;
  Elapsed time:  0.00 s
hypertable> drop namespace test;
  Elapsed time:  1.55 s
```

3.2. Table 表

创建表

```
CREATE TABLE logging (
        uuid,
        tag,
        asctime,
        facility,
        priority,
        message,
        operator
)
```

```
hypertable> CREATE TABLE logging (
         -> uuid,
         -> tag,
```

```
         -> asctime,
         -> facility,
         -> priority,
         -> message,
         -> operator
         -> );

  Elapsed time:  2.14 s
```

列出所有表

```
hypertable> show tables;
logging

  Elapsed time:  0.00 s
hypertable>
```

显示创建该表的HQL

```
hypertable> show create table logging;

CREATE TABLE logging (
  uuid,
  tag,
  asctime,
  facility,
  priority,
  message,
  operator,
  ACCESS GROUP default (uuid, tag, asctime, facility, priority,
message, operator)
)
```

```
  Elapsed time:  0.00 s
hypertable>
```

删除表

```
hypertable> drop table logging;

  Elapsed time:  1.33 s
```

4. FAQ

4.1. 切换 DFS Broker

从local切换到ceph

```
/opt/hypertable/current/bin/stop-servers.sh ;
/opt/hypertable/current/bin/start-dfsbroker.sh local ;
/opt/hypertable/current/bin/clean-database.sh
rm -rf /opt/hypertable/current/hyperspace/*
/opt/hypertable/current/fs/*
/opt/hypertable/current/run/rsml_backup/*
/opt/hypertable/current/run/last-dfs
```

启用ceph

```
# /opt/hypertable/current/bin/start-dfsbroker.sh ceph
DFS broker: available file descriptors: 1024
Waiting for DFS Broker (ceph) (localhost:38030) to come up...
```

第 18 章 CouchBase

Membase + CouchDB = CouchBase, CouchBase是Membase + CouchDB 两个项目合并而来。

1. 安装 CouchBase

进入 http://www.couchbase.com/download 找到适合你的版本，然后使用yum install 安装， 我个人习惯使用yum而不是rpm，因 为 yum 可以解决包之间的依赖问题。

```
# yum install
http://packages.couchbase.com/releases/2.2.0/couchbase-server-
community_2.2.0_x86_64.rpm
```

CouchBase 安装后会自动启动起来，同时启动脚本也做了设置

```
# chkconfig couchbase-server --list
couchbase-server        0:off   1:off   2:on    3:on    4:on
5:on    6:off
```

Web 管理界面http://localhost:8091/index.html

1.1. Getting Started with Couchbase on PHP

安装C开发包

```
# wget -O/etc/yum.repos.d/couchbase.repo
http://packages.couchbase.com/rpm/couchbase-centos62-
x86_64.repo
# yum install -y libcouchbase-devel
```

安装PHP扩展

```
# pecl search couchbase
Retrieving data...0%
Matched packages, channel pecl.php.net:
=======================================
Package   Stable/(Latest) Local
couchbase 1.2.2 (stable)        Couchbase Server PHP extension


#  pecl install couchbase
```

配置扩展

```
cat > /srv/php/etc/conf.d/couchbase.ini <<EOF
extension=couchbase.so
EOF
```

测试代码

```
<?php
// adjust these parameters to match your installation
$cb = new Couchbase("127.0.0.1:8091", "", "", "default");
$cb->set("a", 101);
var_dump($cb->get("a"));
?>
```

```
# php test.php
int(101)
```

2. couchbase 命令

2.1. couchbase-cli

```
couchbase-cli server-list -c 192.168.0.1:8091 -u Administrator
-p password --output=json
```

```
# couchbase-cli server-list -c 192.168.2.16:8091 -u
Administrator -p password
ns_1@127.0.0.1 192.168.2.16:8091 healthy active
```

第 19 章 Memcached

1. 安装 Memcached

1.1. CentOS 下编译

libevent

```
# yum install libevent libevent-devel -y
```

memcache

```
# wget http://memcached.googlecode.com/files/memcached-
1.4.5.tar.gz
# tar zxf memcached-1.4.5.tar.gz
# cd memcached-1.4.5
# ./configure --prefix=/usr/local/memcached-1.4.5
# make && make install
```

start

```
# ln -s /usr/local/memcached-1.4.5 /usr/local/memcached
# /usr/local/memcached/bin/memcached -d -m 128 -p 11211 -u
nobody -l 172.16.0.1
```

1.2. Ubuntu 下编译安装

http://www.monkey.org/~provos/libevent/

```
cd /usr/local/src/
wget http://www.monkey.org/~provos/libevent-1.4.13-
stable.tar.gz
tar zxf libevent-1.4.13-stable.tar.gz
cd libevent-1.4.13-stable
./configure --prefix=/usr/local/libevent-1.4.13-stable
make
make install
make verify

ln -s /usr/local/libevent-1.4.13-stable /usr/local/libevent
ln -s /usr/local/libevent/lib/* /usr/lib/
ln -s /usr/local/libevent/include/* /usr/include/
ln -s /usr/local/libevent/lib/* /usr/local/lib/
ln -s /usr/local/libevent/include/* /usr/local/include/
```

http://www.danga.com/memcached/

```
cd /usr/local/src/
wget http://memcached.googlecode.com/files/memcached-
1.4.5.tar.gz
tar zxf memcached-1.4.5.tar.gz
cd memcached-1.4.5
./configure --prefix=/usr/local/memcached-1.4.5 --with-
libevent=/usr/local/libevent
make
make install

ln -s /usr/local/memcached-1.4.5/ /usr/local/memcached
ln -s /usr/local/memcached/bin/memcached /usr/sbin/memcached
```

/usr/local/memcached/bin/memcached -d -m 2048 -l 127.0.0.1 -p 11211 -u root -c 15000 -P /tmp/memcached.pid

例 19.1. /etc/init.d/memcached

```
#!/bin/bash
# memcached init file for memcached
```

```
#
# chkconfig: - 100 100
# description: a distributed memory object caching system
# author: Neo Chen<openunix@163.com>
#
# processname: /usr/sbin/memcached
# config:
# pidfile: /var/run/memcached

# source function library
. /etc/init.d/functions

OPTIONS="-d -m 2048 -l 127.0.0.1 -p 11211 -u root -c 4096 -P
/var/run/memcached"
USER=daemon
RETVAL=0
prog="memcached"

start() {
        echo -n $"Starting $prog: "
        if [ $UID -ne 0 ]; then
                RETVAL=1
                failure
        else
                daemon --user=$USER /usr/sbin/memcached
$OPTIONS
                RETVAL=$?
                [ $RETVAL -eq 0 ] && touch
/var/lock/subsys/memcached
        fi;
        echo
        return $RETVAL
}

stop() {
        echo -n $"Stopping $prog: "
        if [ $UID -ne 0 ]; then
                RETVAL=1
                failure
        else
                killproc /usr/sbin/memcached
                RETVAL=$?
                [ $RETVAL -eq 0 ] && rm -f
/var/lock/subsys/memcached
        fi;
```

```
        echo
        return $RETVAL
}

reload(){
        echo -n $"Reloading $prog: "
        killproc /usr/sbin/memcached -HUP
        RETVAL=$?
        echo
        return $RETVAL
}

restart(){
        stop
        start
}

condrestart(){
    [ -e /var/lock/subsys/memcached ] && restart
    return 0
}

case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  restart)
        restart
        ;;
#  reload)
#       reload
#        ;;
  condrestart)
        condrestart
        ;;
  status)
        status memcached
        RETVAL=$?
        ;;
  *)
        echo $"Usage: $0
{start|stop|status|restart|condrestart}"
```

```
        RETVAL=1
esac

exit $RETVAL
```

/etc/init.d/memcached

```
chmod +x /etc/init.d/memcached
```

flush_all指令清空memcache中的数据

```
$ telnet 172.16.3.51 11511
Trying 172.16.3.51...
Connected to 172.16.3.51.
Escape character is '^]'.
flush_all
OK
quit
Connection closed by foreign host.
```

1.3. debian/ubuntu

```
$ sudo apt-get install memcache
```

/etc/memcached.conf

```
$ cat /etc/memcached.conf
# memcached default config file
# 2003 - Jay Bonci <jaybonci@debian.org>
# This configuration file is read by the start-memcached script
provided as
# part of the Debian GNU/Linux distribution.
```

```
# Run memcached as a daemon. This command is implied, and is
not needed for the
# daemon to run. See the README.Debian that comes with this
package for more
# information.
-d

# Log memcached's output to /var/log/memcached
logfile /var/log/memcached.log

# Be verbose
# -v

# Be even more verbose (print client commands as well)
# -vv

# Start with a cap of 64 megs of memory. It's reasonable, and
the daemon default
# Note that the daemon will grow to this size, but does not
start out holding this much
# memory
-m 64

# Default connection port is 11211
-p 11211

# Run the daemon as root. The start-memcached will default to
running as root if no
# -u command is present in this config file
-u nobody

# Specify which IP address to listen on. The default is to
listen on all IP addresses
# This parameter is one of the only security measures that
memcached has, so make sure
# it's listening on a firewalled interface.
-l 127.0.0.1

# Limit the number of simultaneous incoming connections. The
daemon default is 1024
# -c 1024

# Lock down all paged memory. Consult with the README and
homepage before you do this
# -k
```

```
# Return error when memory is exhausted (rather than removing
items)
# -M

# Maximize core file limit
# -r
```

restart

```
$ sudo /etc/init.d/memcached restart
```

1.4. yum install

CentOS 6.x

```
# yum install memcached
# chkconfig memcached on
# chkconfig --list memcached

# cat /etc/sysconfig/memcached
PORT="11211"
USER="memcached"
MAXCONN="1024"
CACHESIZE="64"
OPTIONS=""

# /etc/init.d/memcached start
Starting memcached:                                        [
OK  ]
```

CentOS 7.x

```
[root@netkiller ~]# yum install -y memcached
[root@netkiller ~]# rpm -ql memcached.x86_64
/etc/sysconfig/memcached
/usr/bin/memcached
/usr/bin/memcached-tool
/usr/lib/systemd/system/memcached.service
/usr/share/doc/memcached-1.4.15
/usr/share/doc/memcached-1.4.15/AUTHORS
/usr/share/doc/memcached-1.4.15/CONTRIBUTORS
/usr/share/doc/memcached-1.4.15/COPYING
/usr/share/doc/memcached-1.4.15/ChangeLog
/usr/share/doc/memcached-1.4.15/NEWS
/usr/share/doc/memcached-1.4.15/README.md
/usr/share/doc/memcached-1.4.15/protocol.txt
/usr/share/doc/memcached-1.4.15/readme.txt
/usr/share/doc/memcached-1.4.15/threads.txt
/usr/share/man/man1/memcached-tool.1.gz
/usr/share/man/man1/memcached.1.gz

[root@netkiller ~]# cat /etc/sysconfig/memcached
PORT="11211"
USER="memcached"
MAXCONN="1024"
CACHESIZE="64"
OPTIONS=""

[root@netkiller ~]# systemctl enable memcached
Created symlink from /etc/systemd/system/multi-
user.target.wants/memcached.service to
/usr/lib/systemd/system/memcached.service.

[root@netkiller ~]# systemctl start memcached

[root@netkiller ~]# systemctl status memcached
● memcached.service - Memcached
   Loaded: loaded (/usr/lib/systemd/system/memcached.service;
enabled; vendor preset: disabled)
   Active: active (running) since Wed 2018-07-04 11:28:42 CST;
5s ago
 Main PID: 4186 (memcached)
   CGroup: /system.slice/memcached.service
           └─4186 /usr/bin/memcached -u memcached -p 11211 -m
64 -c 1024
```

```
Jul 04 11:28:42 netkiller systemd[1]: Started Memcached.
Jul 04 11:28:42 netkiller systemd[1]: Starting Memcached...
```

2. Memcached 代理

2.1. moxi

couchbase 使用 moxi 为用户提供 memcached 负载均衡功能

2.2. memagent

第 20 章 RethinkDB

http://www.rethinkdb.com/

第 21 章 TokyoCabinet/Tyrant

http://www.162cm.com/p/tokyotyrant.html

```
# yum install tokyocabinet tokyocabinet-devel -y
```

第 22 章 Flare

第 23 章 Voldemort

第 24 章 LevelDB

LevelDB is a fast key-value storage library written at Google that provides an ordered mapping from string keys to string values.

http://code.google.com/p/leveldb/

第 25 章 HyperDex

http://hyperdex.org/

第 26 章 LeoFS

LeoFS is an Unstructured Object Storage for the Web and a highly available, distributed, eventually consistent storage system.