学习目标

今日学习目标

```
1 Redis的应用
    Redis的应用场景、定位
    redis软件的基本使用
    Reids在web项目中的使用
    redis的复制和集群
2 keepalived
    集群的三种表现样式及其特点
    网站的稳定性指标有哪些
    keepalived的基本原理
    keepalived的基本实践
        lvs的DR模型 + Nginx 来进行
```

环境

```
Redis 环境
    如果你的电脑资源有限制的话，一台主机模拟redis伪集群(端口号)
    上课环境：三台主机，实践redis集群(ip地址)
    资源：(2G左右)
keepalived环境
    四台主机，配置不需要太多(2G左右)
```

额外需求

```
学习完毕redis之后，
    1 通过Dockerfile 或者 docker commit 来实现 redis基本环境
    2 基于Dockerfile的使用原则，将 redis的复制集群+sentinel集群 实现出来
```

Redis

基础知识

数据分类

小结

结构化数据

1 数据存储本身是有意义的 -- 强关联和存储约束

2 数据存储的整体，在业务场景中也有关联

-- 一对多、多对一、一对一等

半结构化数据

1 开发场景：

页面的展示 和 展示的数据 分开

数据本身没有意义，但是组合在一起能够使用

- json

2 测试场景；

自动化测试 -- 构造大量的测试数据单独保存

- xml

非结构化数据

数据本身存储和业务场景存储没有所谓的关联

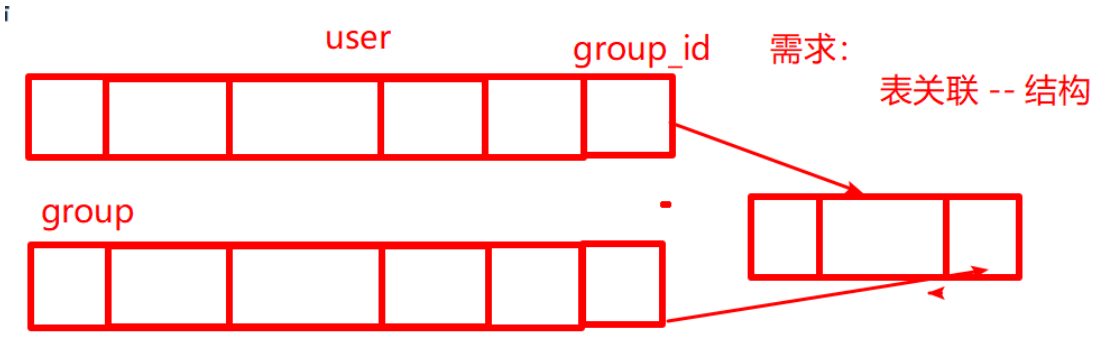
-- 但是我要的时候，必须给我

kv

注意：

这里的结构 不是 数据结构与算法里面 数据存储应用到的结构

指的是 业务场景中的数据关联存储



```
{
  "status": 200,
  "message": {
    "person": [
      {
        "id": 1,
        "name": "张三",
        "gender": "男",
        "address": {
          "Country": "中国",
          "Province": "北京市",
          "city": "北京市",
          "district": "丰台区",
          "town": "五里店"
        },
      },
    ],
  }
}
```

用户名

性别

籍贯

```
<?xml version="1.0" encoding="gb2312"?>
<namelist>
  <name1>
    <ID>01</ID>
    <name>张三</name>
    <sex>男</sex>
    <address>北京市市丰台区五里店</address>
  </name1>
<namelist>
```

数据关系

存储格式是以节点为主,一个节点衍生出另外的子节点

每个节点遵循html的风格，但是里面的标签属性是我们自定义的。

用户名

性别

地址

登录 注册

数据软件的应用

目前 结构化软件还是占据绝对的位置

非结构化数据软件，在特定场景中，占据一席地位

目前所有的软件，都有一个趋同的发展

我是结构化数据软件，但是能做对方 的事情。

Nosql

小结

简介

NoSQL -- Not only SQL

常见的软件：

Redis、MongoDB、Hbase

数据的存储样式：

SQL - 二维表样式

Nosql

本质上都是以 k/v 样式来存储，但是在应用的时候，各有特点

web开发场景以json为主

自动化测试场景，以 xml 为主(我说的)

存储理论

小结

前提：

一台主机的资源，无法抗住大量用户数据的操作，所以我们需要以集群的方式，来对数据进行管理

问题：

如果保证 集群主机 数据是一致的，对用户来说无所谓

话题：

事务

在一些业务场景中，一个操作需要多个sql才能够完成指定的功能

-- 这个整体操作就是一个事务

ACID

A 原子性 -- 对于事务操作来说(多条命令)，要么成功，要么一起失败还原

B 一致性 -- 事务操作前 和后 ，对于数据库本身的数据访问功能没有影响

C 隔离性 -- 同一个数据集群内部的多个事务操作，彼此间无交叉影响

D 持久性 -- 数据的落地

CAP

前提：

在分布式集群场景中，无法做到单台主机能够实现的 ACID，那么就做一个缓冲

C

-- 集群间所有主机数据是一致的

-- 数据库集群的同步

A -- 集群整体提供的服务对用户来说，可用

--

P -- 集群提供服务的过程中，允许出现一些错误数据，或者过期数据，

-- 访问课程数据的时候，运行是一个月前的过期数据

```
BASE
    前提：
        我已经确定了，集群环境中，不可能不存异常故障，接下来只能从
            CAP 里面的 C一致性 和 A可用性 之间来找平衡

    BA   基本可用
        -- 无论任何时候，我们的网站服务是正常，虽然效果没有预期的那么好
    S    软状态
        -- 针对的是 集群内部的主机 状态转换的时候 -- 一个中间过渡
    E    最终一致性
        -- 即使集群内部出现故障了，但是最终故障恢复后，要与其他主机数据进行同步
```

Redis

小结

```
简介
    1 Redis 开源的，基于内存的数据库
    2 Redis 支持很多种(>5)数据类型存储
    3 Redis 支持各种功能
            - 复制、内部检测、事务操作、数据持久化、高可用功能(高可用、高扩展)

应用场景
    以数据存储本身的角度来说场景
        有序集合     - 各种排行、topn
        list        - 数据的排布，顺序
        sort集合     - 范围数据列表
        string      - 数据的存储
        hash字典     - 数据分类(子分类)
```

redis部署

小结

```
redis 安装
    apt方式
    源码安装

命令
    systemctl restart redis-server
```

```
    redis-server /path/to/redis.conf [--port 启动端口]
    注意：
        一条命令-个配置文件可以启动多个redis节点

    redis-cli 进入默认的redis服务
    redis-cli -h 主机ip -p 主机

    redis-cli shutdown
        -- 整体使用
    redis-cli -h 主机ip -p 主机 shutdown
        -- 这个命令，在redis 伪集群中使用

配置文件：
    文件名： redis.conf
    内容组成：25个部分配置
    常用的：
        MODULES
        general、network、snapshottiing、APPEND ONLY MODE
        集群相关-但是与课程无关
        REDIS CLUSTER等
```

默认开启的配置文件属性

```
root@python-auto:~# grep -Env '#|^$' /etc/redis/redis.conf
75:bind 127.0.0.1 -::1                                                  *
94:protected-mode yes
98:port 6379
*****
107:tcp-backlog 511
119:timeout 0
136:tcp-keepalive 300
257:daemonize yes                                                       *
275:supervised auto
289:pidfile /run/redis/redis-server.pid                                 *
297:loglevel notice
302:logfile /var/log/redis/redis-server.log                             *
327:databases 16
*****
336:always-show-logo no
341:set-proc-title yes
358:proc-title-template "{title} {listen-addr} {server-mode}"
398:stop-writes-on-bgsave-error yes
404:rdbcompression yes
413:rdbchecksum yes
```

```
431:dbfilename dump.rdb
****
444:rdb-del-sync-files no
454:dir /var/lib/redis
****
510:replica-serve-stale-data yes
526:replica-read-only yes
555:repl-diskless-sync no
567:repl-diskless-sync-delay 5
593:repl-diskless-load disabled
627:repl-disable-tcp-nodelay no
668:replica-priority 100
878:acllog-max-len 128
1123:lazyfree-lazy-eviction no
1124:lazyfree-lazy-expire no
1125:lazyfree-lazy-server-del no
1126:replica-lazy-flush no
1133:lazyfree-lazy-user-del no
1140:lazyfree-lazy-user-flush no
1207:oom-score-adj no
1217:oom-score-adj-values 0 200 800
1230:disable-thp yes
1252:appendonly no
1256:appendfilename "appendonly.aof"
1282:appendfsync everysec
1304:no-appendfsync-on-rewrite no
1323:auto-aof-rewrite-percentage 100
1324:auto-aof-rewrite-min-size 64mb
1348:aof-load-truncated yes
1359:aof-use-rdb-preamble yes
1377:lua-time-limit 5000
1573:slowlog-log-slower-than 10000
1577:slowlog-max-len 128
1598:latency-monitor-threshold 0
1649:notify-keyspace-events ""
1714:hash-max-ziplist-entries 512
1715:hash-max-ziplist-value 64
1730:list-max-ziplist-size -2
1746:list-compress-depth 0
1753:set-max-intset-entries 512
1758:zset-max-ziplist-entries 128
1759:zset-max-ziplist-value 64
1773:hll-sparse-max-bytes 3000
1783:stream-node-max-bytes 4096
1784:stream-node-max-entries 100
```

```
1804:activerehashing yes
1839:client-output-buffer-limit normal 0 0 0
1840:client-output-buffer-limit replica 256mb 64mb 60
1841:client-output-buffer-limit pubsub 32mb 8mb 60
1872:hz 10
1888:dynamic-hz yes
1894:aof-rewrite-incremental-fsync yes
1900:rdb-save-incremental-fsync yes
2019:jemalloc-bg-thread yes
```

简单实践

```
进入redis
    redis-cli                                进入默认的redis服务
    redis-cli -h 主机ip -p 主机              标准的入口
    redis-cli --raw                          中文调试场景

查看效果
    info    查看redis的全部信息
    ping    检测主机存活效果
    set     设置一个值
    get     获取一个值
    help    帮助命令
```

基本命令

基本操作

小结

```
redis命令帮助
127.0.0.1:6379> help
redis-cli 6.2.5
To get help about Redis commands type:
        查看redis组命令帮助 -- 命令集合
      "help @<group>" to get a list of commands in <group>
        查看具体的命令
      "help <command>" for help on <command>
        自动输出相关命令列表 -- 命令组和内置的一些简单命令
      "help <tab>" to get a list of possible help topics
        退出redis环境
      "quit" to exit

To set redis-cli preferences:
```

```
        设置(命令)命中的效果
      ":set hints" enable online hints
        取消(命令)命中的效果
      ":set nohints" disable online hints
Set your preferences in ~/.redisclirc
    个人的行为习惯属性设置


基础命令
    help            查看帮助命令
    config          临时修改redis.conf 配置属性，重启服务后失效
    info [内容段]    查看系统的运行状态信息
```

基本操作

```
确认数据是否存在
    exists

新增数据
    set key value ex seconds

查看数据
    get key

查看类型
    type key

删除数据
    del key [key .. key]

清空数据库【危险】
    flushdb  -- 清空当前数据库的所有key
    flushall -- 清空当前redis所有库的key
```

String操作

小结

```
特点:
    其他数据类型的数据表现样式
    简单的数据存储
    示例:
        cookie、session、校验码等
```

命令

```
set key value
setex key seconds value
mset key1 value1 key2 value2 。。。

get key
mget key

del key 【key】
```

list

小结

简介

只要是列表场景，或者说符合列表 属性的场景，都可以来使用

命令

```
lpush key value [value]
rpush key value [value]
linsert key before|after 现存值 新增值

lrange key start stop
llen key
lindex key value
lpos key pos_index

lrem key count value
    指定数值来进行删除
        count > 0 从左侧开始删除
        count < 0 从右侧开始删除
        count = 0 全部删除

lpop key count
rpop key count

ltrim key 保留起始索引 结束索引
```

set集合

小结

```
场景：
    内容不重复的任何场景都可以

命令
    sadd key memeber memeber

    smemeber key
    sunion key1 key2
    sinter key1 key2
    sdiff key1 key2
        -- 返回 key1里面特有的一些内容，key2没有

    srem key member
    spop key count
```

有序集合 sortset

```
场景：
    排行榜、topN

命令
    zadd key score member score member

    zrange key 起始 结束
        默认所有的数据是按照score 从小到大来进行排序
        如果后面添加了 rev 就是 从大到小来进行排序
    ZRANGEBYSCORE mysorted 0 -1
        获取指定分数范围的内容
    ZREMRANGEBYSCORE key 66 100
        将符合分数要求内容删除
```

hash字典

小结

```
场景：
    某个对象的特定属性：
        person: {
```

```
            username: zhangsan,
            password: 123456,
            address: beijign,
            xxx: xxx
        }
```

命令

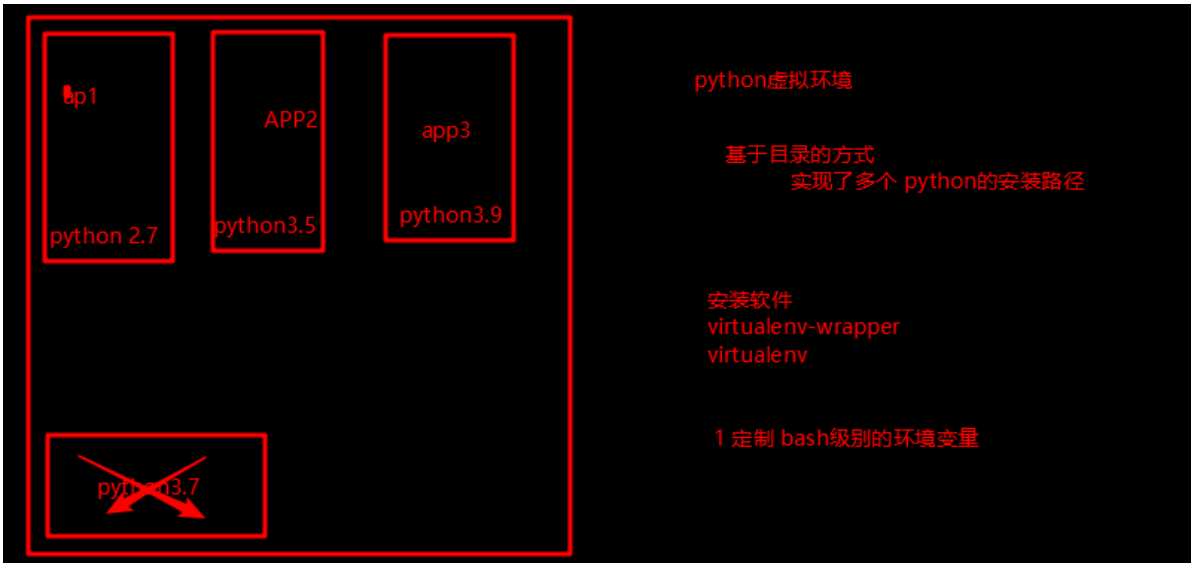
```
    hset key field vallue field value

    hkeys key
        -- 获取key里面所有的 field名
    hmget key field field
        -- 获取指定field里面的 value

    hdel key field
        -- 删除 key里面指定的 field
    del key
```

python实践

前提




```
python@python-auto:~$ tail -n3 .bashrc
# 定制虚拟环境的配置
export WORKON_HOME=$HOME/.virtualenvs
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
```

虚拟环境命令

```
workon   切换到指定的虚拟环境
deactivate 退出虚拟环境
mkvirtualenv   指定python版本创建虚拟环境
rmvirtualenv   删除指定的python版本
```

redis模块

```
模块
    redispy
连接
    redis_obj = redis.Redis(host='ip', port=6379,db=1)

操作:
    redis_obj.set('key', 'value')
    redis_obj.get('key')
    redis_obj.delete('key')
```

flask应用

```
简介
    flask 是一个 基于python的web框架
    注意:
        我们这里仅仅演示一下 flask应用的基本使用,

演示



redis演示:
    1 安装模块
    2 配置flask应用
    3 多个路由
        添加key到 redis
        获取key到 redis
```

持久复制

持久化

小结

```
简介
    redis默认支持两种 基于磁盘的数据持久化技术

原理
    rdb
        优势:
            - 基于数据的快照来进行存储
```

- 数据完整
- 策略非常灵活

劣势：

- 数据量大的时候，快照文件也大
- bgsave的时候，会以覆盖的方式同步数据，有可能导致部分数据丢失
  对于此我们可以借助于 定时备份的方式将数据额外保存

aof

优势：

- 基于操作命令的方式进行数据的存储
- 容量非常小

劣势：

- 对于基础的数据有一定的依赖

使用：

rdb做基础数据的备份

aof做实时数据的备份

主从同步

简介

对于主从同步来说，

主角色不用做任何配置

- 开放自己的怀抱即可

从角色需要做两个方面的配置

1 bind 开放本机的ip

2 replicaof 指定主角色

实践

主角色

数据的增删改查

从角色

从主角色主机里获取数据

数据的查看

特点：

如果从角色主机故障，那么主角色主机中的从主机状态会自动消除

如果主角色主机故障，那么整个集群就崩溃了(相对于数据更改来说)

主角色的状态

```
# Replication
role:master
connected_slaves:1
slave0:ip=10.0.0.13,port=6379,state=online,offset=246,lag=1
master_failover_state:no-failover
master_replid:f9c916d74f4252f0c30c890ecb6a8f4666665f52
master_replid2:0000000000000000000000000000000000000000
master_repl_offset:260                ******
second_repl_offset:-1
repl_backlog_active:1
repl_backlog_size:1048576
repl_backlog_first_byte_offset:1
repl_backlog_histlen:260
```

从角色的状态

```
# Replication
role:slave
master_host:10.0.0.12
master_port:6379
master_link_status:up
master_last_io_seconds_ago:4
master_sync_in_progress:0
slave_repl_offset:347            *****
slave_priority:100
slave_read_only:1
replica_announced:1
connected_slaves:0
master_failover_state:no-failover
master_replid:94d9590b1e8204b0b6d2e36410f7183cea396e21
master_replid2:0000000000000000000000000000000000000000
master_repl_offset:347           *****
second_repl_offset:-1
repl_backlog_active:1
repl_backlog_size:1048576
repl_backlog_first_byte_offset:1
repl_backlog_histlen:347
```

哨兵集群

小结

```
问题：
    主从复制的时候，主故障，整个集群(对于更新操作来说)就崩了
```

需求：
    主从复制集群，即使出现主角色主机故障，也不影响整个redis集群的正常稳定运行

思路：
    1 在所有的redis节点主机上，配备一个 sentinel 节点
    2 所有的 sentinel 节点内部协商默认的主名称 及 默认的ip地址
    3 启动sentinel集群即可

特性：
    监控
        - sentinel 会监控所有的redis的节点信息
    故障切换
        - 如果主redis故障，则被sentinel下线，然后从 slave角色的redis节点中选择新的主
    通知
        - 更换redis主角色主机的时候，会通知其他 slave角色主机更新配置找新主
    配置更改
        - sentinel内部会自动更新相关的配置。

实践

要点：
    前提：
        redis节点间的主从同步效果

    1 配置
        1 bind 本地ip(不要用 127.0.0.1开头)
        2 daemonize yes 以后台的方式来启动
        3 sentinel monitor mymaster 10.0.0.12 6379 2

    2 命令
        redis-sentinel /path/to/sentinel.conf

    3 进入sentinel 界面
        redis-cli -h 10.0.0.12 -p 26379
        info 查看 sentinel的状态
            -- sentinel 主机的数量是3

    4 故障演练
        1 关闭10.0.0.12的redis服务
            查看 10.0.0.12 里面的sentinel的状态
            查看 10.0.0.12 里面的sentinel的日志转换信息
        2 关闭10.0.0.13的redis服务
            查看 10.0.0.12 里面的sentinel的状态
            查看 10.0.0.12 里面的sentinel的日志转换信息

3 依次开启 10.0.0.12 和 10.0.0.13 里面的redis服务
查看 10.0.0.12 里面的sentinel的状态
查看 10.0.0.12 里面的sentinel的日志转换信息

注意：
sentinel 集群
不仅仅会更改sentinel本身的与redis节点的配置信息
也会更改 redis节点本身关于 replicaof的 配置信息

```
root@python-auto:/data/server/sentinel# grep -Evn '#|^$' sentinel.conf
15:bind 127.0.0.1 10.0.0.12
21:port 26379
26:daemonize no
31:pidfile /var/run/redis-sentinel.pid
36:logfile "/data/server/sentinel/sentinel.log"
65:dir /tmp
84:sentinel monitor mymaster 10.0.0.12 6379 2
125:sentinel down-after-milliseconds mymaster 30000
149:acllog-max-len 128
200:sentinel parallel-syncs mymaster 1
225:sentinel failover-timeout mymaster 180000
298:sentinel deny-scripts-reconfig yes
335:SENTINEL resolve-hostnames no
341:SENTINEL announce-hostnames no
```

哨兵启动后的配置

```
root@python-auto:~# grep -Env '#|^$' /data/server/sentinel/sentinel.conf
15:bind 127.0.0.1 10.0.0.12
21:port 26379
26:daemonize no
31:pidfile "/var/run/redis-sentinel.pid"
36:logfile "/data/server/sentinel/sentinel.log"
65:dir "/tmp"
84:sentinel monitor mymaster 10.0.0.12 6379 2
148:acllog-max-len 128
295:sentinel deny-scripts-reconfig yes
332:sentinel resolve-hostnames no
338:sentinel announce-hostnames no
340:protected-mode no
341:user default on nopass ~* &* +@all
342:sentinel myid 03d108c56c9bc5b672e30febb60a9bd42fb0bec5
343:sentinel config-epoch mymaster 0
344:sentinel leader-epoch mymaster 0
345:sentinel current-epoch 0
346:sentinel known-replica mymaster 10.0.0.13 6379
347:sentinel known-replica mymaster 10.0.0.14 6379
```