**Redis**集群 **-** 配置最简单的**Redis**主从

2017年04月08日 10:57:06

阅读数：13370

要求：

1、你已经知道redis是什么

2、熟悉Linux基本操作(CentOS 6.5)

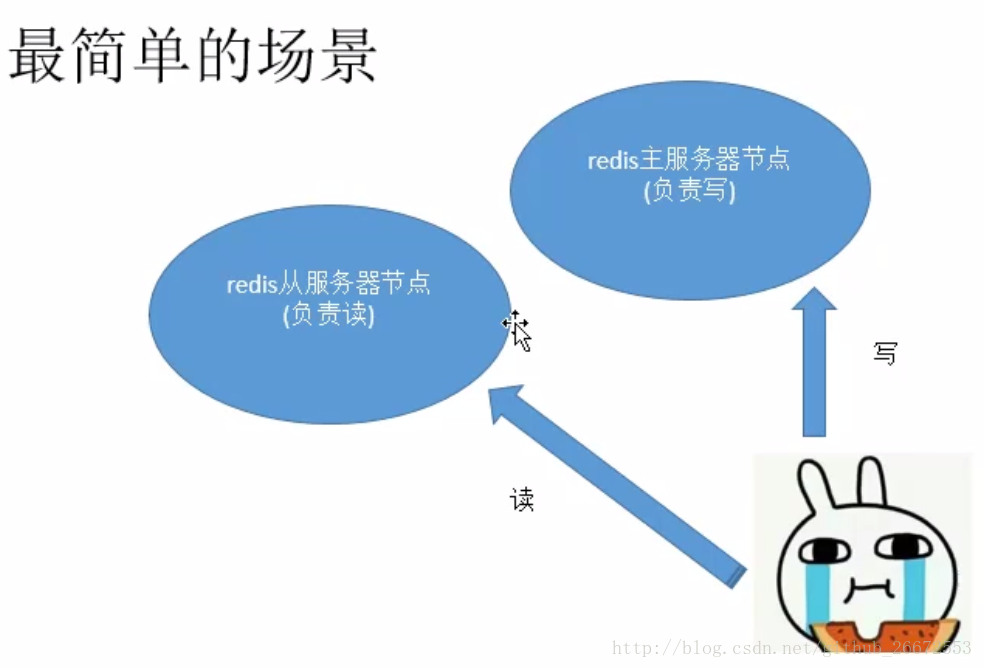
3、有单机redis开发相关经验

4、有主观能动性，至少熟练百度的使用

**redis**主从配置初步：**2**个节点主从

如果我们redis的压力很大，如果我们的并发高到我们读数据和写数据都有了很大压力。

那么我们可能就需要把redis分开部署，并且配置为一个『主从』的状态。

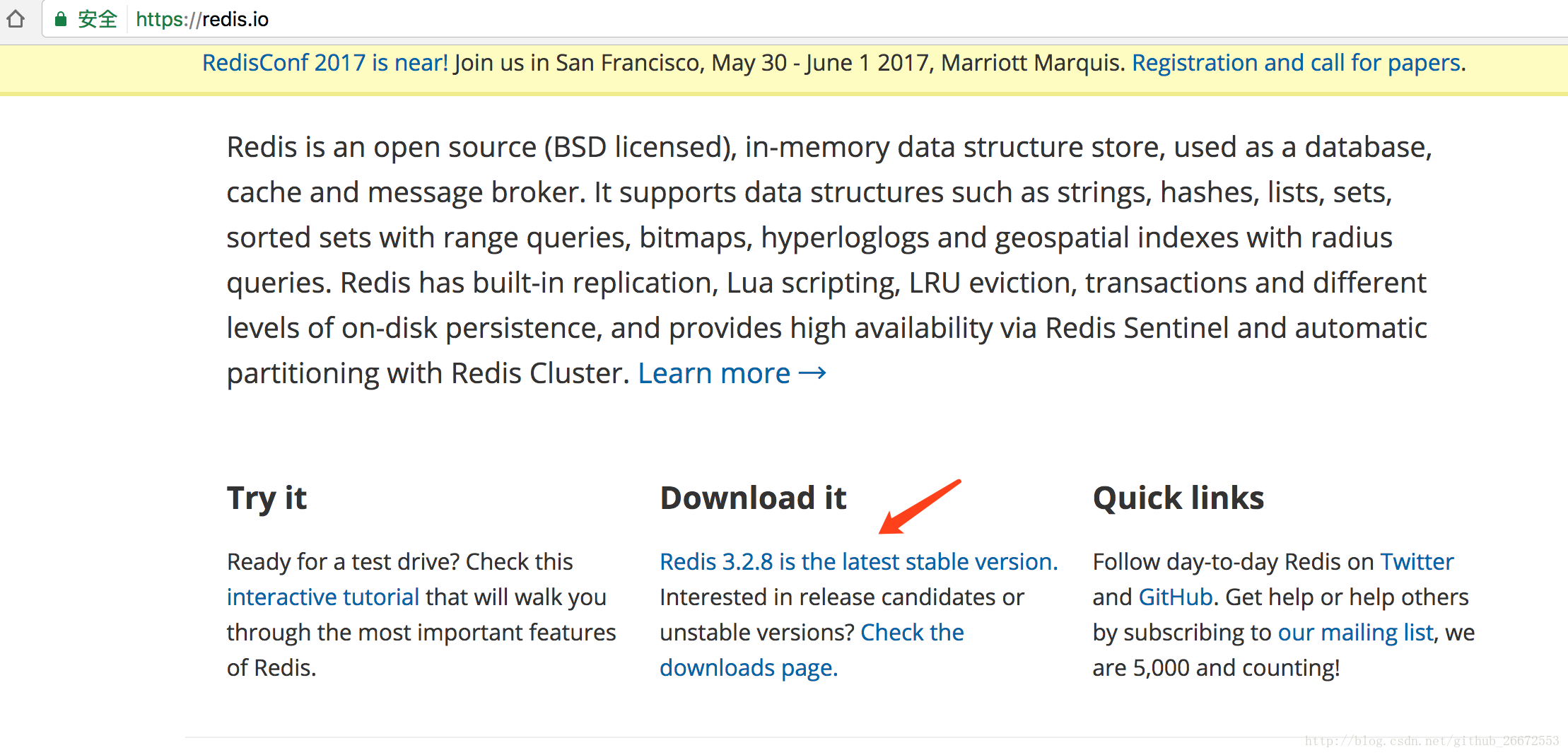


首先我们要安装**redis**

官网地址：

[https://redis.io](https://redis.io/)

(一定要去官网下载)



下载之后，会有一个redis-3.2.8.tar.gz

(注意我们后面整个过程不要使用root帐号，我这里是在一个叫’admin’的用户下’)



1、解压

tar zxvf redis-3.2.8.tar.gz

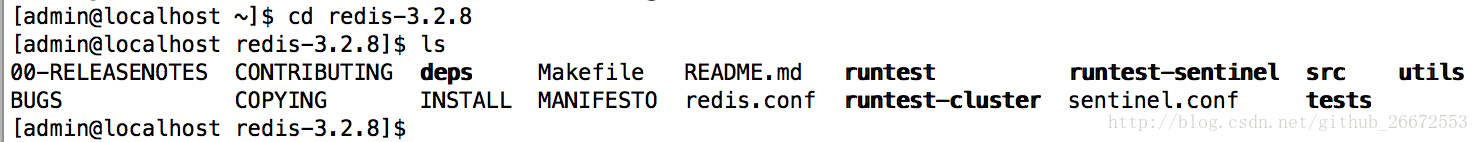
* 1

2、进入redis源码目录

cd redis-3.2.8

* 1

这里边的文件，我们需要编译



3、编译

根据手册

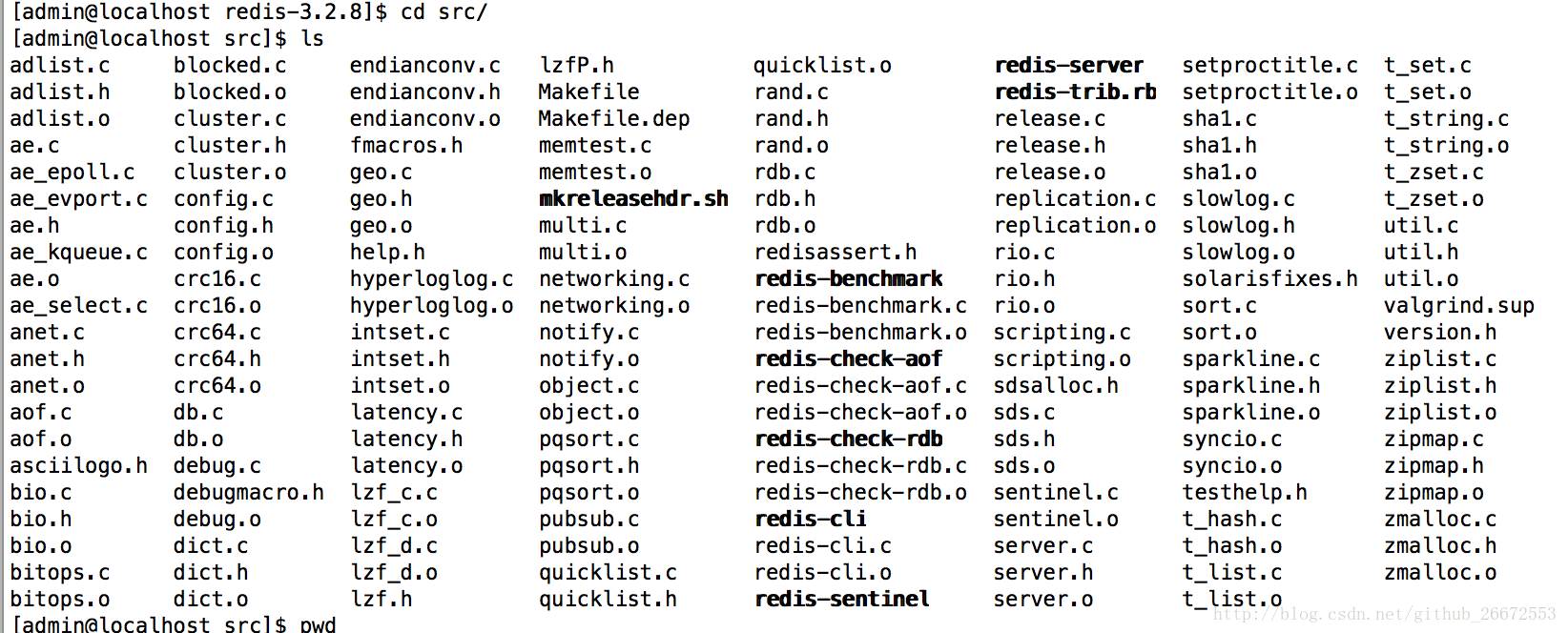
<https://redis.io/download>

执行命令：

make

* 1

编译好后会在src目录生成一些可执行文件



为了方便，我们把一些文件拷贝到**/usr/local/redis**中

1、在/usr/local/下创建需要的目录

cd /usr/local

sudo mkdir redis #创建目录

* 1
* 2

注意：如果报”admin is not in the sudoers file. This incident will be reported.”错误。是因为执行sudo命令的用户不在sudoers文件的列表中。可以通过编辑sudoers文件，来解决这个问题。

切换到root帐号执行：vi /etc/sudoers

为了让admin这个用户能够执行sudo命令，在sudoers文件中加入一行：

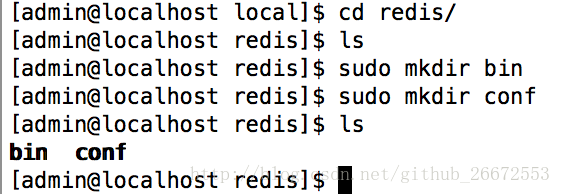
admin ALL=(ALL) ALL

2、/usr/local/redis目录创建成功，需要修改redis目录所有者

sudo chown -R admin:admin redis/

* 1

3、然后在redis目录，新建bin和conf目录。(不是必须的，只不是过为了方便而已)



4、把前面编译后的redis的可执行文件(在redis-3.2.8/src/下)，复制到/usr/local/redis/bin里面去

sudo cp redis\* /usr/local/redis/bin/

#拷贝redis开头的所有文件

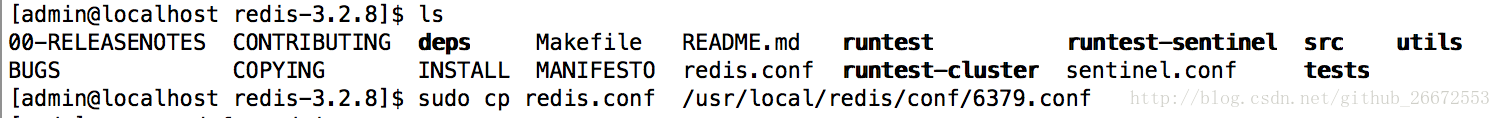
* 1
* 2

5、redis提供给我们了一个默认的配置文件redis.conf

把它复制到/usr/local/redis/conf目录下，并改名为6379.conf

sudo cp redis.conf /usr/local/redis/conf/6379.conf

* 1



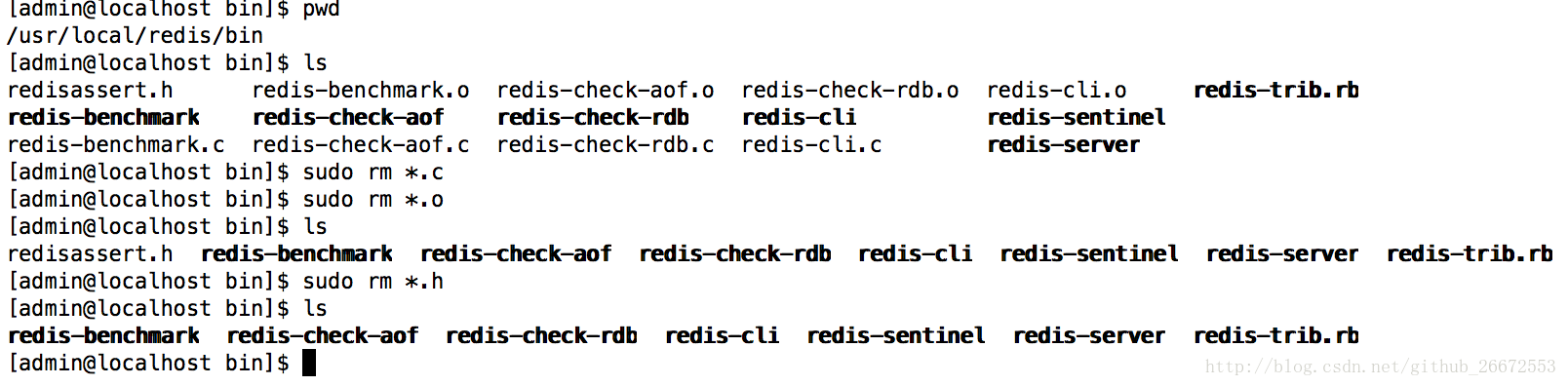
6、删除一些不必要的文件（我们只需要redis的可执行文件）

sudo rm \*.c

sudo rm \*.o

sudo rm \*.h

* 1
* 2
* 3



7、修改配置文件

sudo vi 6379.conf

* 1

修改如下几个配置：

daemonize no

修改为：

daemonize yes (后台程序方式运行)

pidfile /var/run/redis\_6379.pid

修改为：

pidfile /usr/local/redis/redis\_6379.pid

(把pidfile生成到有权限的目录下)

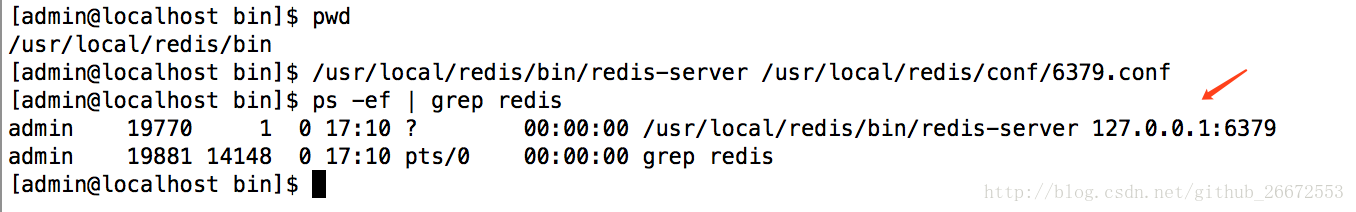
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9

8、启动

注意文件夹路径不要搞错

/usr/local/redis/bin/redis-server /usr/local/redis/conf/6379.conf

* 1



9、配置从节点

sudo cp 6379.conf 6380.conf

* 1

修改6380.conf

修改对应的端口和pid配置

然后加入slaveof 127.0.0.1 6379

port 6379

修改为：

port 6380

pidfile /usr/local/redis/redis\_6379.pid

修改为：

pidfile /usr/local/redis/redis\_6380.pid

增加一行：

slaveof 127.0.0.1 6379

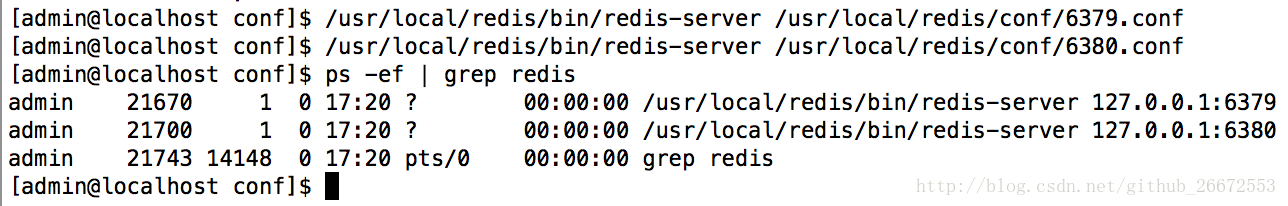
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10

10、启动2个redis实例

/usr/local/redis/bin/redis-server /usr/local/redis/conf/6379.conf

/usr/local/redis/bin/redis-server /usr/local/redis/conf/6380.conf

* 1
* 2



11、启动redis客户端

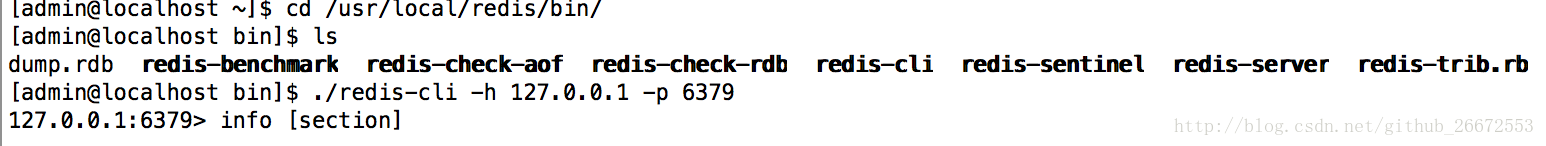
去连接6379那个实例

cd /usr/local/redis/bin/

./redis-cli -h 127.0.0.1 -p 6379

* 1
* 2
* 3

连上之后输入：info命令



可以从中看到很多信息，比如：

role:master #本机是『主』

connected\_slaves:1 #有1个『奴隶』

slave0:ip=127.0.0.1,port=6380,state=online,offset=11957,lag=1

#"从"服务器iP地址和端口是

* 1
* 2
* 3
* 4



12、新增

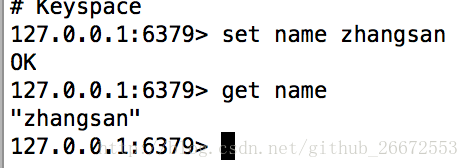
set name zhangsan

* 1

#读取

get name

* 1
* 2

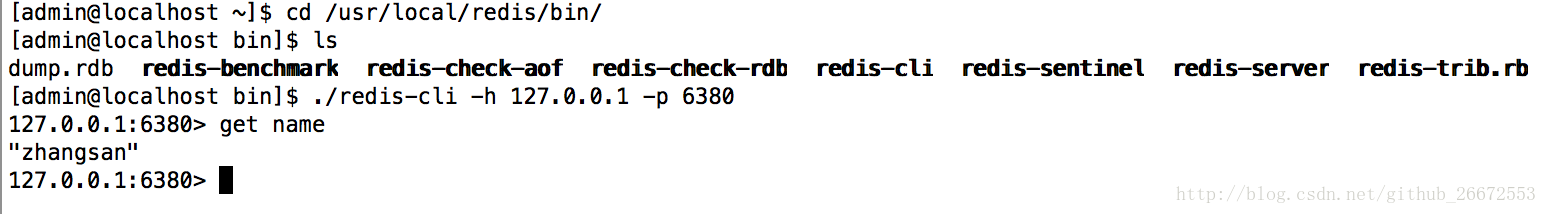


13、然后访问6380那个实例

./redis-cli -h 127.0.0.1 -p 6380

get name

* 1
* 2
* 3



我们发现这2个实例已经完成了数据的同步。

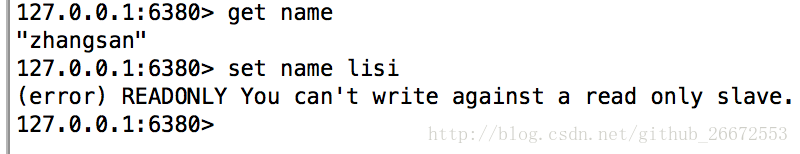
如果我们要在从服务器写入

set name lisi

#会提示：(error) READONLY You can't write against a read only slave.

#因为从服务器只有读权限，我们做的就是redis的读写分离啊

* 1
* 2
* 3



我们这里演示是在一台机器上，分了2个端口，真实开发肯定是2台机器了，不然分离就没有意义了。

——————————————————————————————

**Redis**主从配置异常解决：**Error condition on socket for SYNC: Connection refused**

在docker中搭建的redis主从集群时，从服务器上的redis日志报错：

32677:S 08 Feb 16:14:38.947 \* Connecting to MASTER 172.168.10.70:6379

32677:S 08 Feb 16:14:38.948 \* MASTER <-> SLAVE sync started

32677:S 08 Feb 16:14:38.948 # Error condition on socket for SYNC: Connection refused

32677:S 08 Feb 16:14:39.950 \* Connecting to MASTER 172.168.10.70:6379

32677:S 08 Feb 16:14:39.950 \* MASTER <-> SLAVE sync started

32677:S 08 Feb 16:14:39.950 # Error condition on socket for SYNC: Connection refused

32677:S 08 Feb 16:14:40.952 \* Connecting to MASTER 172.168.10.70:6379

32677:S 08 Feb 16:14:40.952 \* MASTER <-> SLAVE sync started

32677:S 08 Feb 16:14:40.953 # Error condition on socket for SYNC: Connection refused

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9

解决方案：

在redis主服务器上的redis.conf中修改bind字段，将

bind 127.0.0.1

* 1

修改为

bind 0.0.0.0

* 1

又或者直接注释掉bind字段

# bind 127.0.0.1

* 1

原因：

如果redis主服务器绑定了127.0.0.1，那么跨服务器IP的访问就会失败，从服务器用IP和端口访问主的时候，主服务器发现本机6379端口绑在了127.0.0.1上，也就是只能本机才能访问，外部请求会被过滤，这是linux的网络安全策略管理的。如果bind的IP地址是172.168.10.70，那么本机通过localhost和127.0.0.1、或者直接输入命令redis-cli登录本机redis也就会失败了。只能加上本机ip才能访问到。

所以，在研发、测试环境可以考虑bind 0.0.0.0，线上生产环境建议绑定IP地址。