1、普通连接

public class JedisTest {

public static void main(String[] args) {

// 获取jedis实例，要输入主机的ip和端口号

Jedis jedis = new Jedis("192.168.1.29", 6379);

//使用redis命令，创建k-v

System.out.println(jedis);

jedis.set("k2", "123123");

//使用redis命令，查询所有的k

Set<String> keys = jedis.keys("\*");

//输出所有的k

keys.forEach(set -> System.out.println(set));

//redis事务

Transaction transaction = jedis.multi();

transaction.set("k3", "v3");

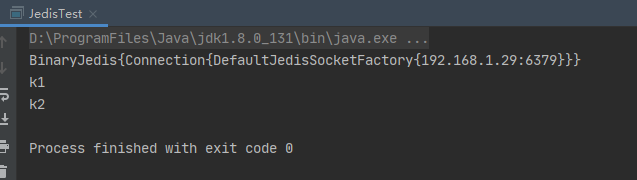
transaction.exec();

//关闭连接

jedis.close();

}

}



Failed to create socket.异常

解决方案：

ip和端口号可能不对

关闭linux防火墙 systemctl stop firewalld.service

2、Redis连接池

public class RedisTest2 {

public static void main(String[] args) {

//构建连接池配置信息

JedisPoolConfig jedisPoolConfig = new JedisPoolConfig();

//设置最大连接数

jedisPoolConfig.setMaxTotal(50);

//构建连接池

String host = "192.168.1.29";

int port = 6379;

JedisPool jedisPool = new JedisPool(jedisPoolConfig, host, port);

//从连接池获取信息

Jedis jedis = jedisPool.getResource();

//读取数据

String val = jedis.get("k1");

System.out.println(val);

//将连接归还到连接池

//jedis3之前使用jedisPool.returnResource(jedis);

//jedis3之后使用jedis.close()

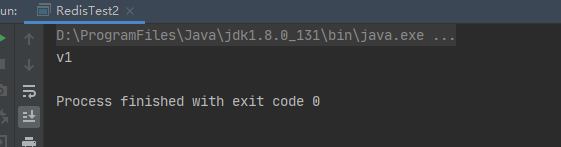
jedis.close();

//释放连接池

jedisPool.close();

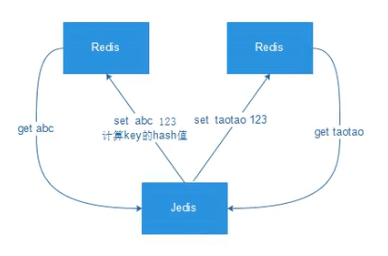
}

}





3、分片式集群



存在的问题：无法动态增加减少服务节点。

/\*\*

\* 分片式集群

\*/

public class JedisTest3 {

public static void main(String[] args) {

//构建连接池配置信息

GenericObjectPoolConfig poolConfig = new JedisPoolConfig();

//JedisPoolConfig poolConfig = new JedisPoolConfig();

//设置最大连接数

poolConfig.setMaxTotal(50);

//定义集群信息

List<JedisShardInfo> shardInfos = new ArrayList<>();

shardInfos.add(new JedisShardInfo("192.168.1.29", 6379));

shardInfos.add(new JedisShardInfo("192.168.1.29", 6380));

//定义集群连接池

ShardedJedisPool shardedJedisPool = new ShardedJedisPool(poolConfig, shardInfos);

ShardedJedis shardedJedis = null;

try {

//从连接池获取到jedis分片对象

shardedJedis = shardedJedisPool.getResource();

//插入数据

for (int i = 0; i < 50; i++) {

shardedJedis.set("key\_" + i, "val\_" + i);

}

//从redis获取数据

String v1 = shardedJedis.get("key\_44");

String v2 = shardedJedis.get("key\_16");

System.out.println(v1);

System.out.println(v2);

} catch (Exception e) {

e.printStackTrace();

} finally {

if (null != shardedJedis) {

//关闭，检测连接是否有效，有效则放回连接池，无效则重置状态

shardedJedis.close();

}

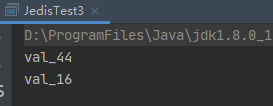
}

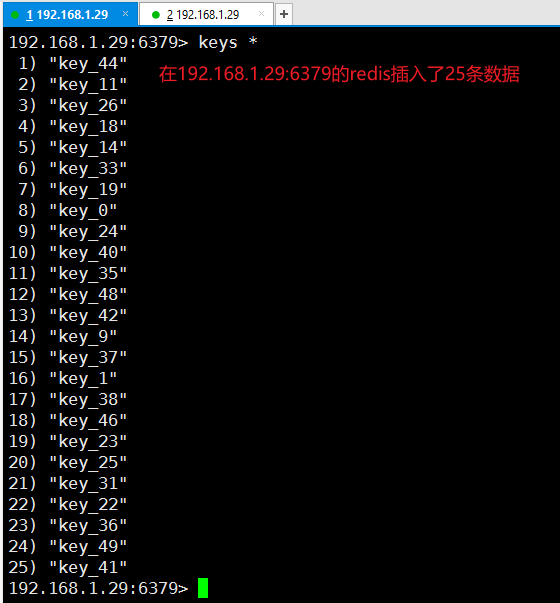
//关闭连接池

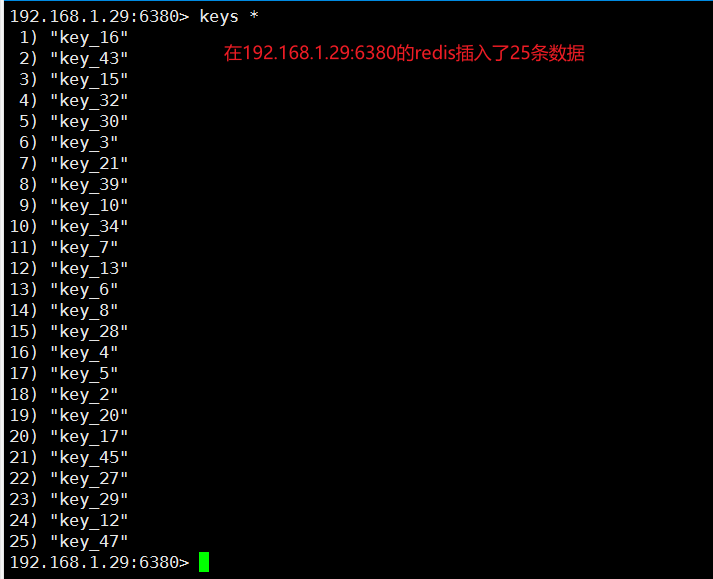
shardedJedisPool.close();

}

}







1. Spring整合Redis