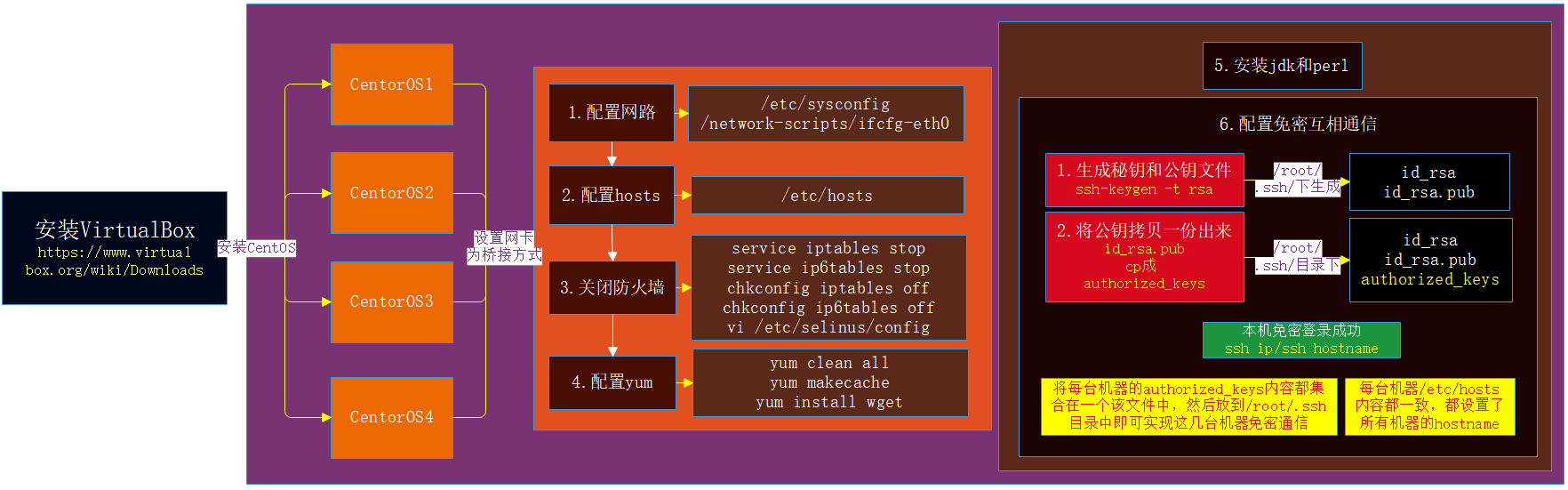
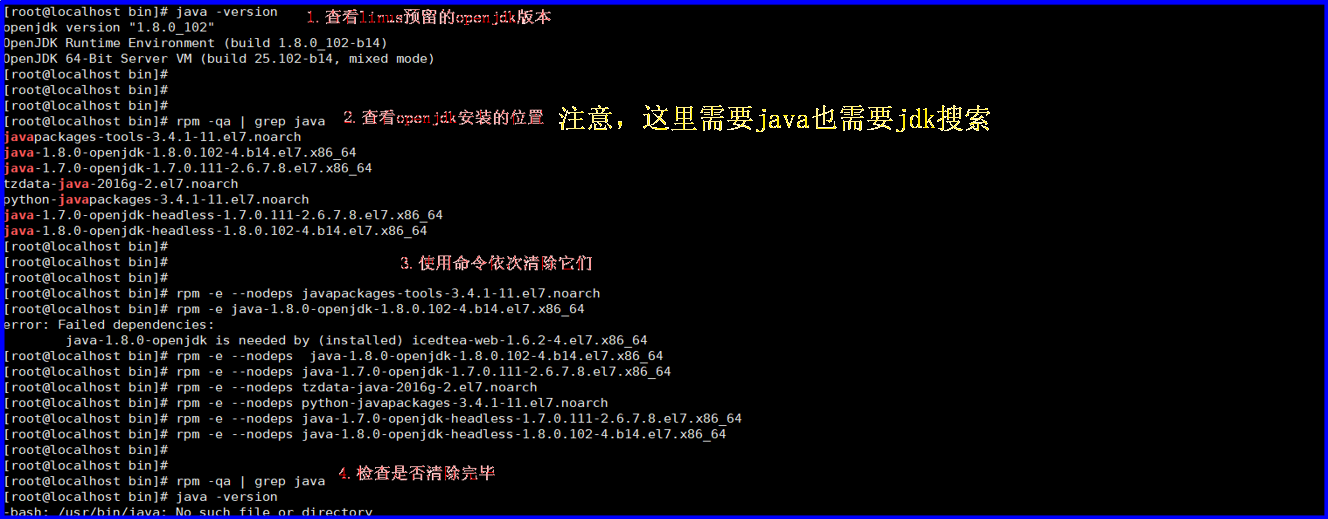
# 从零在虚拟机搭建一个4个节点的CenterOS集群



## 在每个CentorOS中安装jdk和perl

### 安装jdk

（1）卸载linus中自带的openjdk



（2）下载rmp的jdk包并安装

//其中i-install表示安装

//v-verbose安装过程详细信息

//h-hash表示显示进度

rpm -ivh jdk-7u65-linux-i586.rpm

（3）配置jdk环境变量

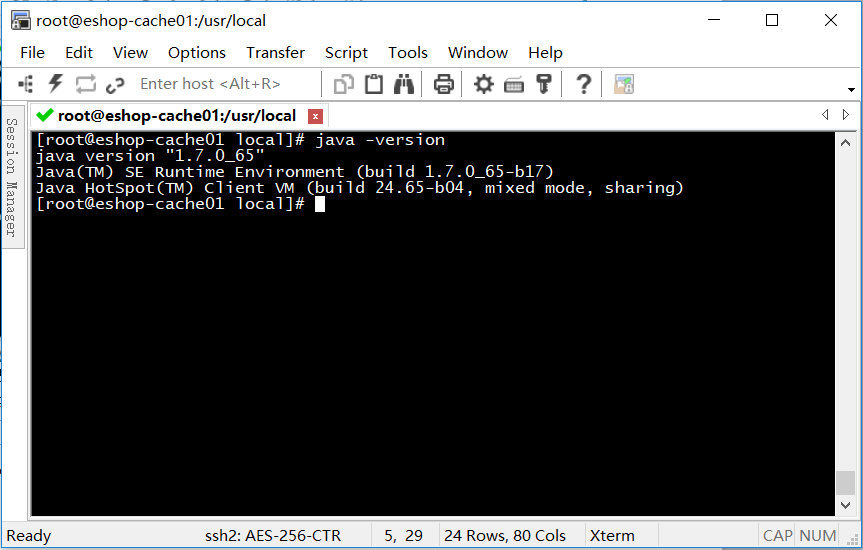
vi ~/.bashrc //find / -name .bashrc,寻找.bashrc文件位置

export JAVA\_HOME=/usr/java/latest

export PATH=$PATH:$JAVA\_HOME/bin

source ~/.bashrc //退出文件执行source

bash 是一个能解释你输入进终端程序的东西，并且基于你的输入来运行命令。它在一定程度上支持使用脚本来定制功能，这时候就要用到 .bashrc 了。为了加载你的配置，bash 在每次启动时都会加载 .bashrc 文件的内容。每个用户的 home 目录都有这个 shell 脚本。它用来存储并加载你的终端配置和环境变量。终端配置可以包含很多不同的东西。最常见的，.bashrc 文件包含用户想要用的别名。别名允许用户通过更短的名字或替代的名字来指向命令，对于经常在终端下工作的人来说这可是一个省时利器。



### 安装perl

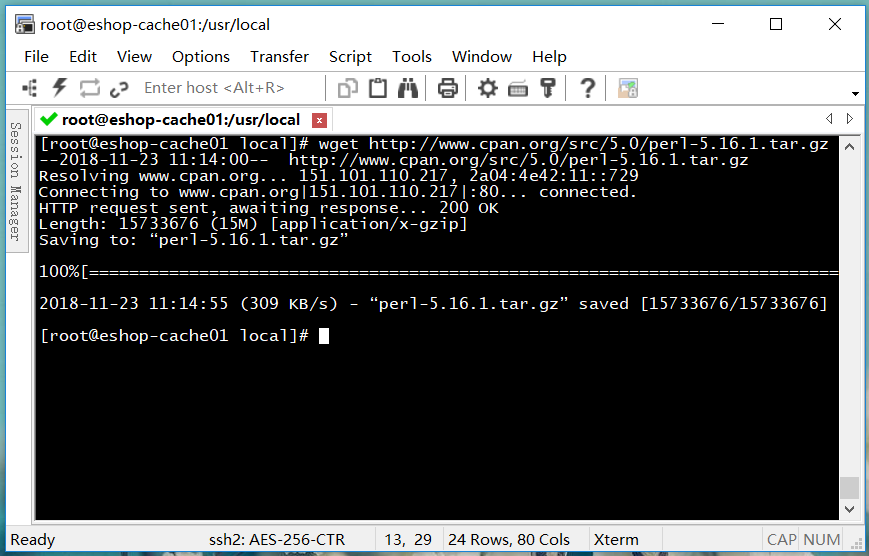
（1）为什么要装perl

整个大型电商网站的详情页系统比较复杂;java+nginx+lua需要perl，它是一个基础的编程语言的安装，tomcat跑java web应用需要用到。

（2）安装perl前的准备工作

//1.利用wget命令下载perl安装包并解压

wget <http://www.cpan.org/src/5.0/perl-5.16.1.tar.gz>



//解压perl压缩包并删除tar.gz压缩包

//tar -zxvf 文件.tar.gz -C /path

//z-gunzip表示该文件是被gunzip解压的,具有gzip属性的

//x-extract表示将文件提取出来

//v-verbose显示解压过程中的一些信息

//f-file指定解压的文件,后面必须跟文件名

//-C 指定解压到文件的位置

tar -zxvf perl-5.16.1.tar.gz

rm -rf perl-5.16.1.tar.gz

//2.安装语言编译库gcc

yum -y install gcc //耗时较长

（3）安装perl

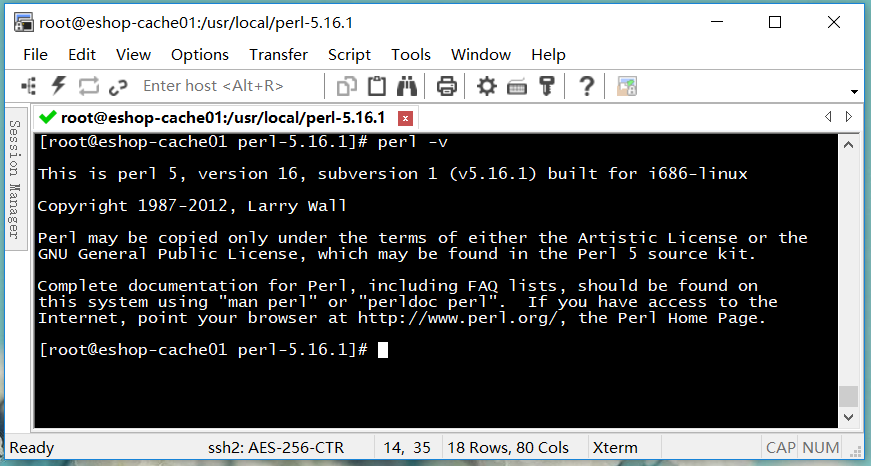
cd /usr/local/perl-5.16.1//进入per解压后的目录

./Configure -des -Dprefix=/usr/local/perl

make && make test && make install //耗时很长

（4）测试perl是否装好

perl -v

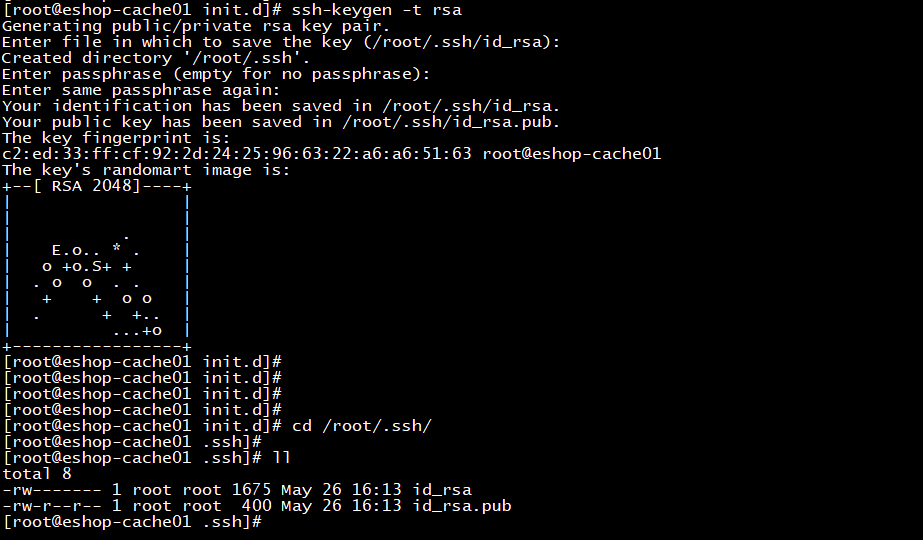


## 为4台CentOS配置ssh免密互相通信

### 生成本机公钥

//过程中不断敲回车,公钥放在/root/.ssh目录下

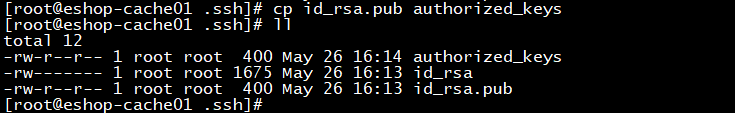
ssh-keygen -t rsa



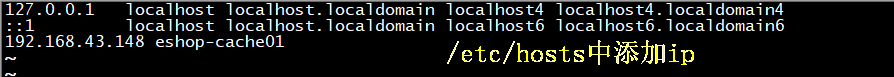
### 将公钥cp一份为authorized\_keys

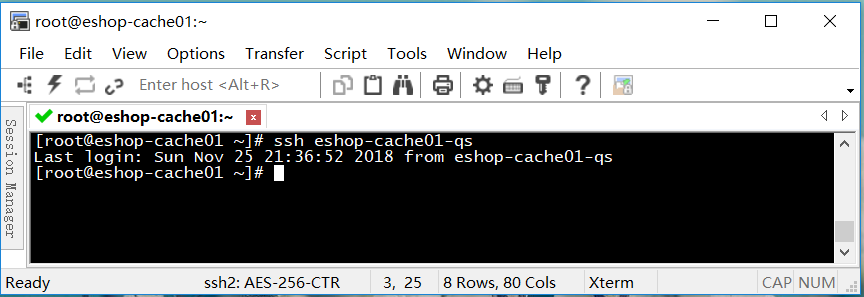
cd /root/.ssh

cp id\_rsa.pub authorized\_keys



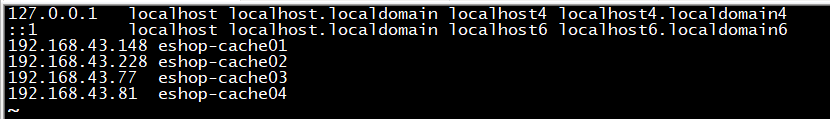
此时ssh以ip免密码可以直接登录，还需要在每台机器中的/etc/hosts中配置所有电脑的ip name，以ssh hostname登录更方便，如下图所示：

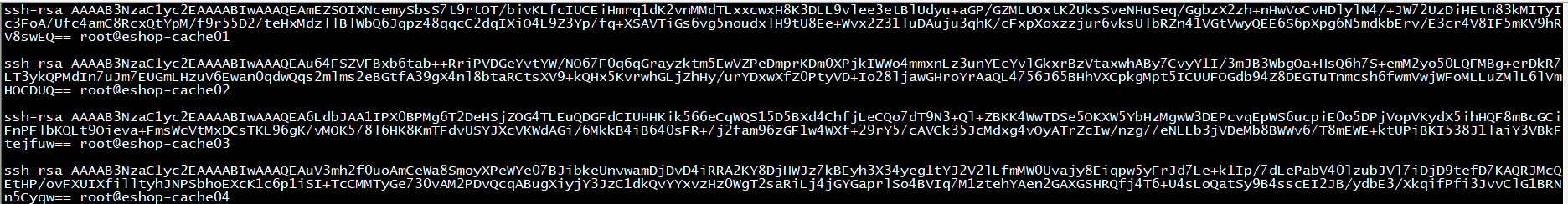




### 配置三台机器之间ssh免密码登录

保证每台机器/etc/hosts和/root/authorized\_keys(四台机器综合)内容一致，其中authorized\_keys是四台机器各自生成并copy公钥内容的总和





### /root/.ssh/中文件作用

（1）authorized\_keys：

里面记录了每台机器的访问权限信息-公钥-id\_rsa.pub,有其他机器的公钥时，其他机器访问本机器时无需密码;

（2）id\_rsa:

本机器的私钥,在执行命令 ssh-keygen -t rsa时和公钥id\_rsa.pub一起生成;

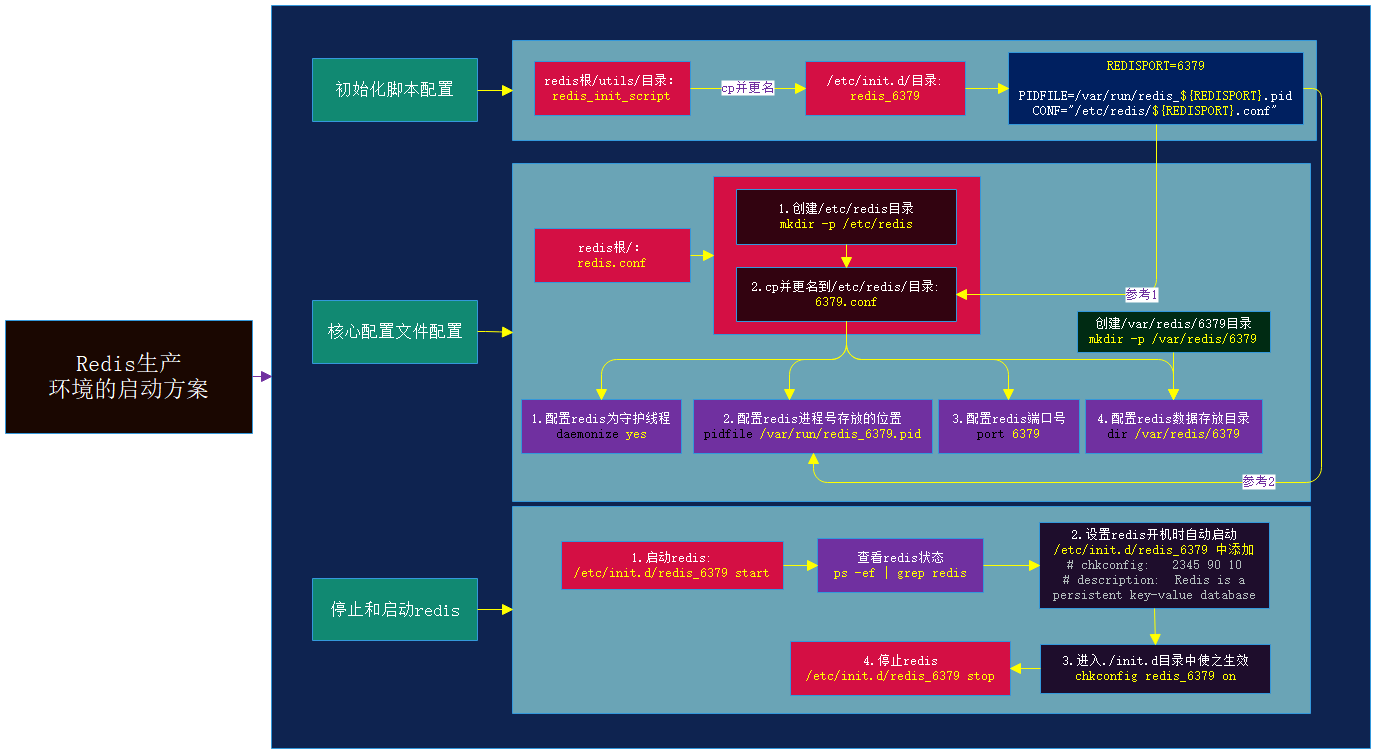
（3）id\_rsa.pub:

本机器的公钥，需要将它cp为authorized\_keys，方便本机器访问本机器，并且便于其他机器作为识别是否有本机器权限的一个唯一标识;

（4）know\_hosts:

会记录本机器访问过其他哪些机器的公钥，如果没有记录，第一次会询问yes/no，若选择yes，则会在文件中记录访问机器的公钥;

# Redis单机版的安装及生产环境启动方案



## Redis单机版的安装

### 下载并安装tcl

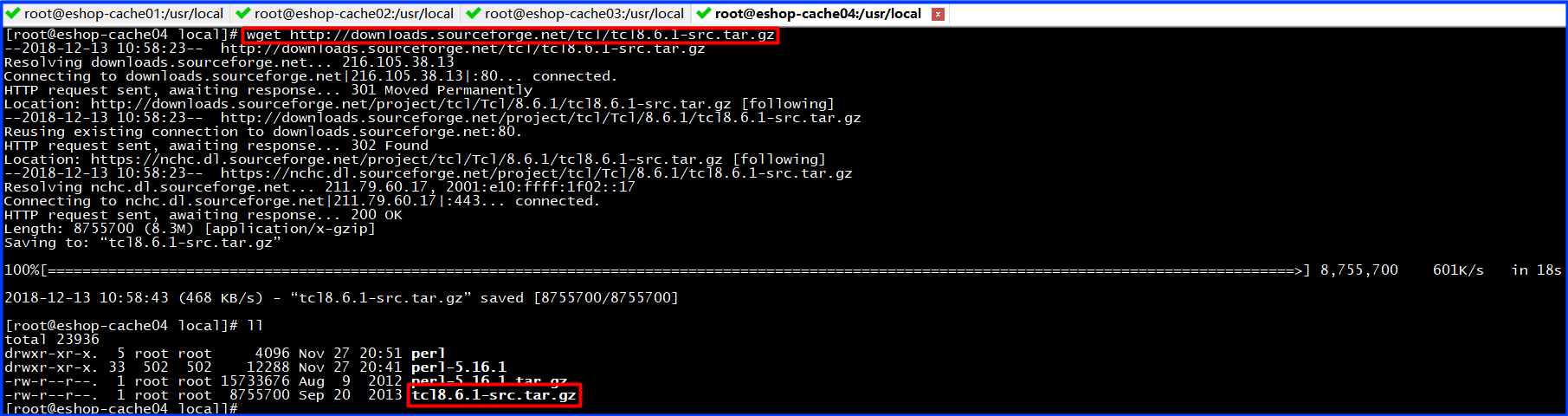
1、下载wget

wget 是一个从网络上自动下载文件的自由工具，支持通过 HTTP、HTTPS、FTP 三个最常见的 [TCP/IP协议](https://baike.baidu.com/item/TCP/IP%E5%8D%8F%E8%AE%AE) 下载，并可以使用 HTTP 代理。"wget" 这个名称来源于 “World Wide Web” 与 “get” 的结合。



2、下载tcl

wget http://downloads.sourceforge.net/tcl/tcl8.6.1-src.tar.gz



3、安装tcl

1):解压tcl

tar -zxvf tcl8.6.1-src.tar.gz

2):执行.configure文件

cd /usr/local/tcl8.6.1/unix/

./configure

3):安装tcl-漫长

cd /usr/local/tcl8.6.1/unix

make && make test && make install

### 安装redis

1、准备redis的安装包

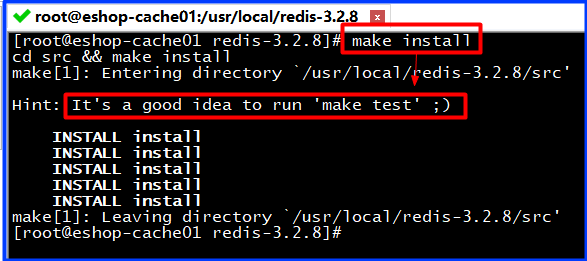


2、解压并安装-漫长

tar -zxvf redis-3.2.8.tar.gz

cd /usr/local/redis-3.2.8

make && make test && make install



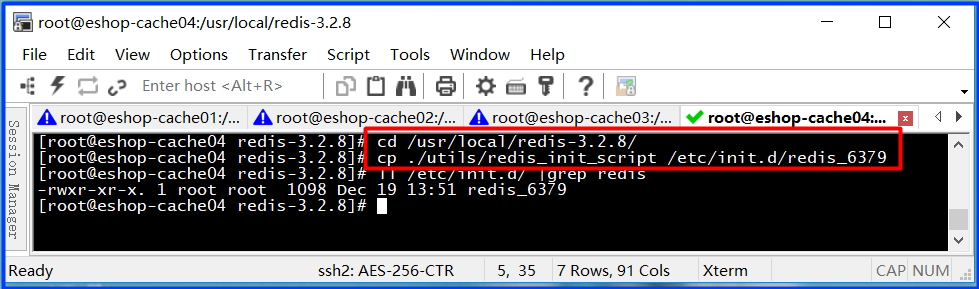
1. 再次在../redis-3.2.8目录下安装make install,否则报如下错误



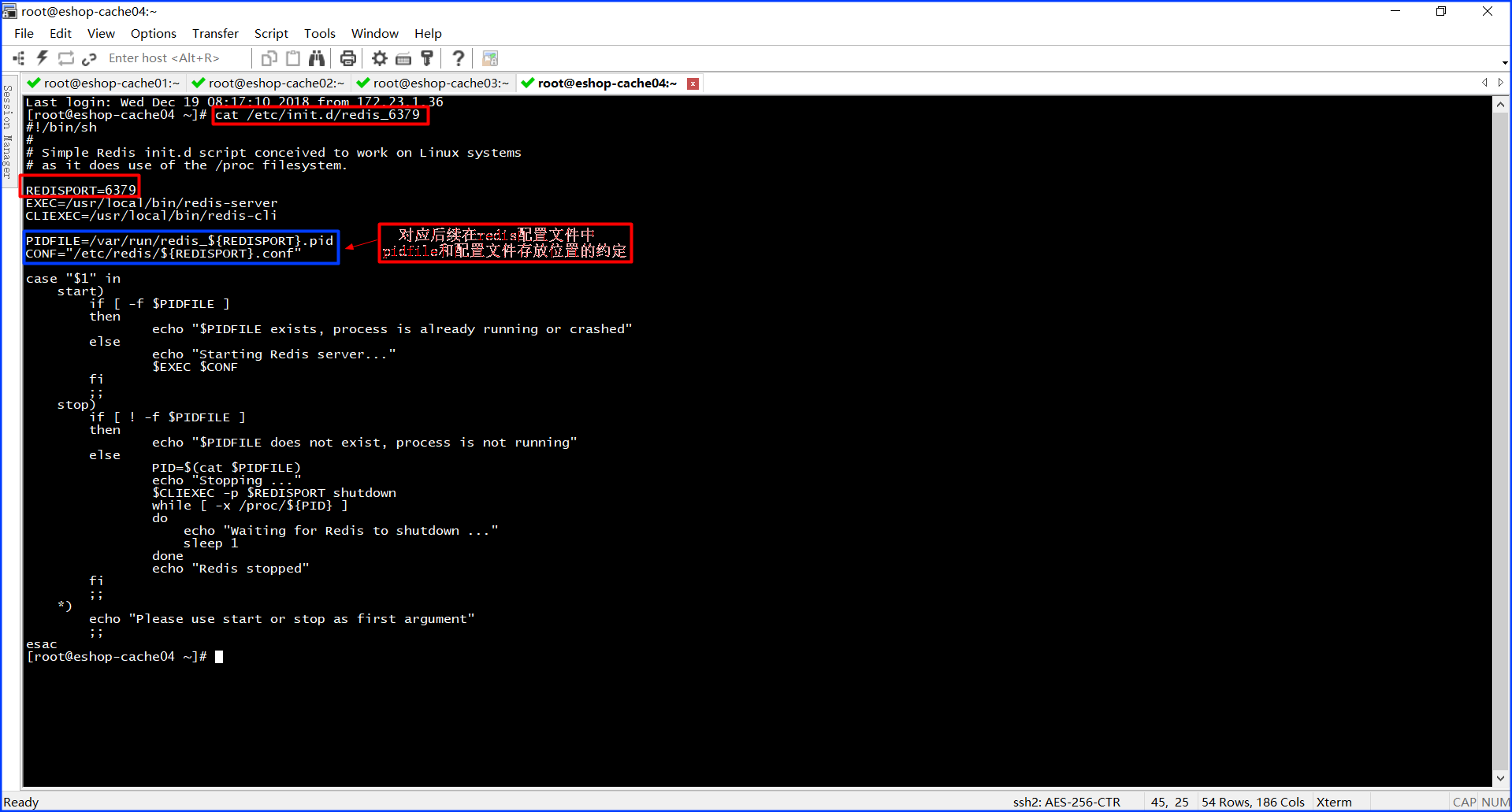
## Redis生产环境的启动方案

### 初始化脚本文件配置

将redis/utils/redis\_init\_script拷贝到/etc/init.d/下,并重命名为redis\_6379

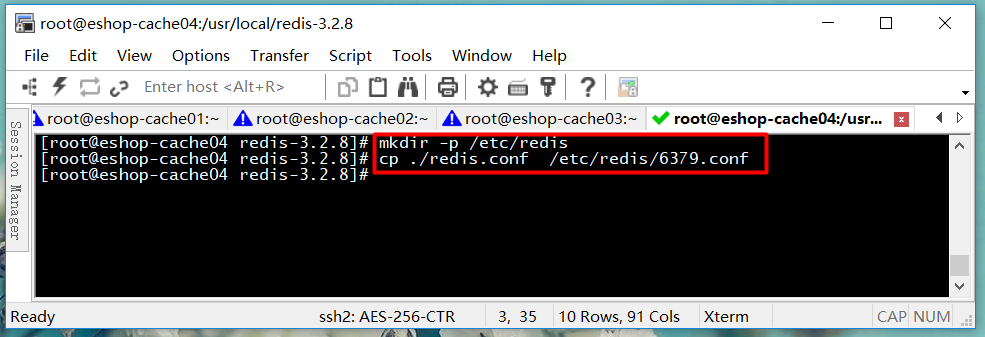


默认redis\_6379文件中的REDISPORT为6379



### 核心配置文件配置

1.将redis根/redis.conf拷贝到/etc/redis/目录下，并重命名为6379.conf



2.配置/etc/redis/6379.conf

1):daemonize yes

使redis以守护进程的方式运行;

2):pidfile /var/run/redis\_6379.pid

启动redis成功后,会在/var/run/目录下生成redis\_6379.pid记录当前进程号;

3):port 6379

redis的默认端口;

4):dir /var/redis/6379

/var/redis/6379该目录需要创建,是redis的数据存储位置;

### 启动和停止redis服务

1.常规启动

1):启动redis:

chmod 777 /etc/init.d/redis\_6379

/etc/init.d/redis\_6379 start

2):查看redis是否启动成功:

//ps-process是linus下最常用的也是非常强大的进程查看命令

//e-environment

//f-format

//grep-global regular expression print

ps -ef | grep redis

2.停止redis服务

/etc/init.d/redis\_6379 stop

### 让redis服务开机自动自动

1.在/etc/init.d/redis\_6379中添加一下两行

# chkconfig: 2345 90 10

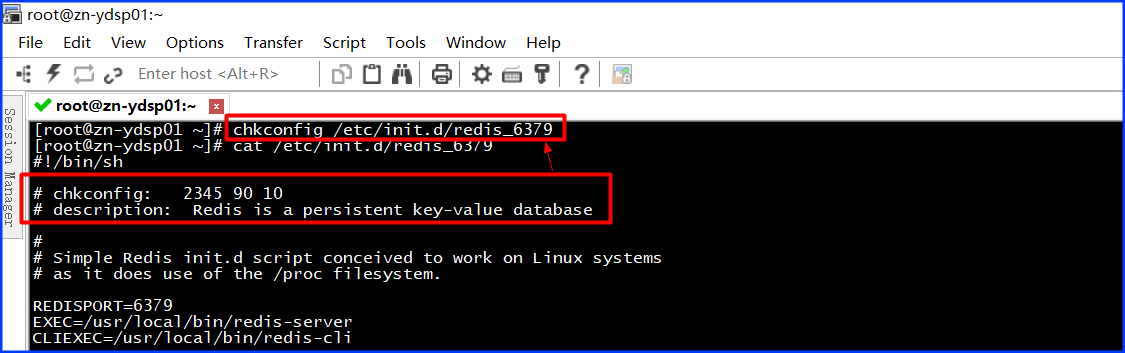
# description: Redis is a persistent key-value database

redis服务必须在运行级2，3，4，5下被启动或关闭，启动的优先级是90，关闭的优先级是10

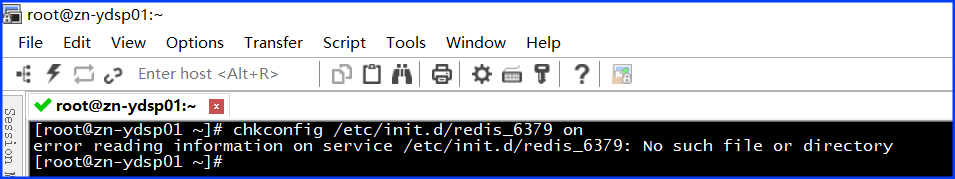
1. 执行如下命令

cd /etc/init.d/

chkconfig redis\_6379 on//此处一定要进入到目录中执行

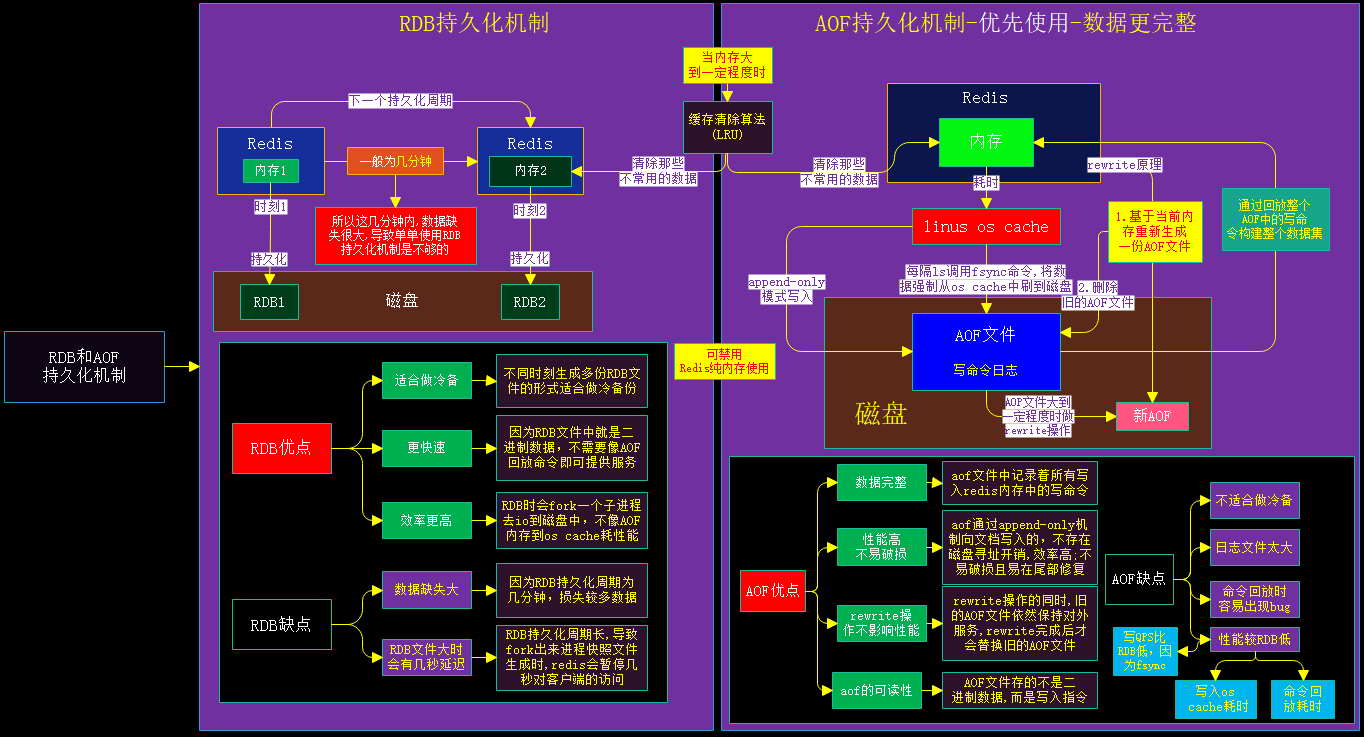


否则报如下错误

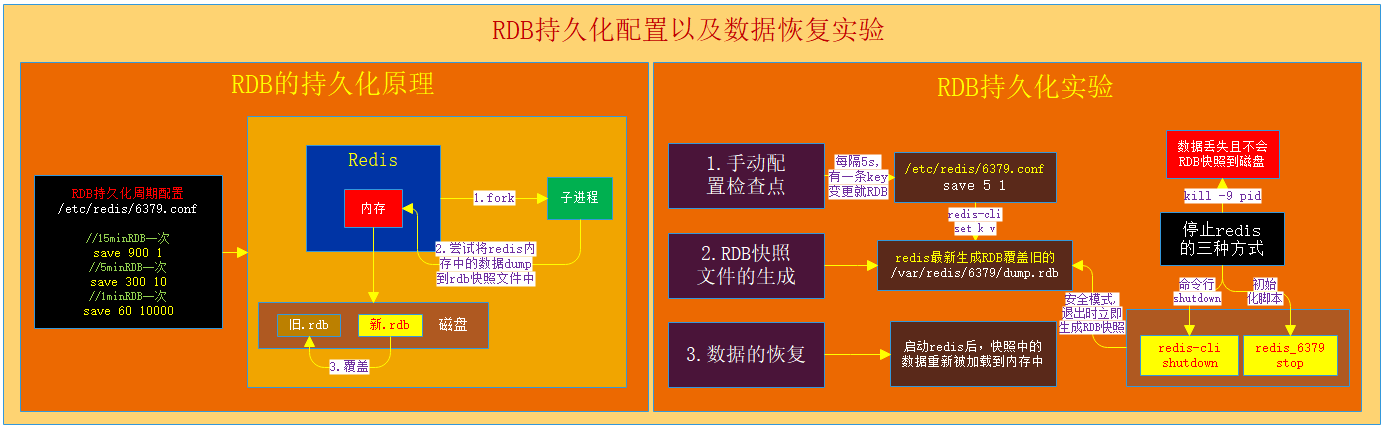


# Redis持久化机制工作原理和企业级数据备份恢复演练

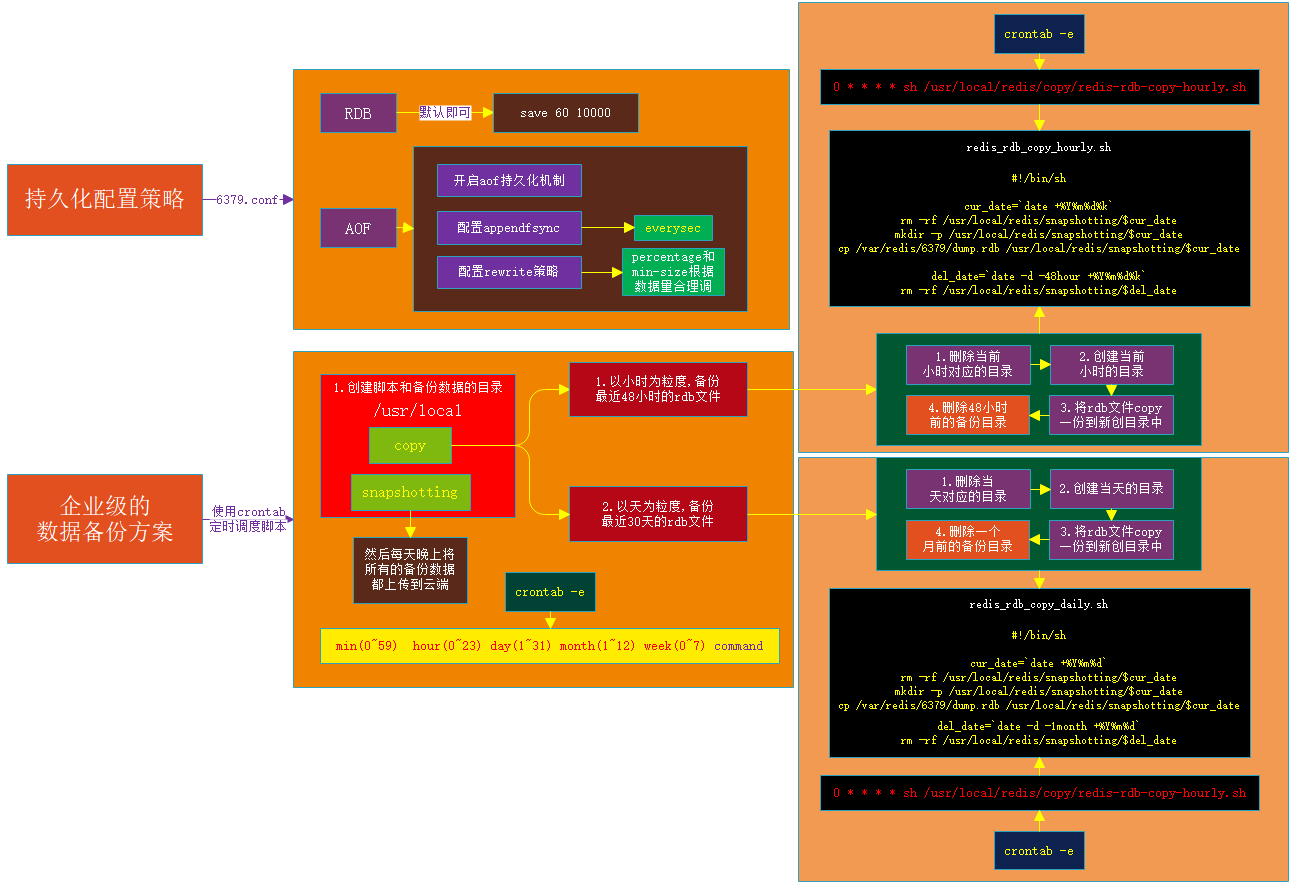
## 原理篇

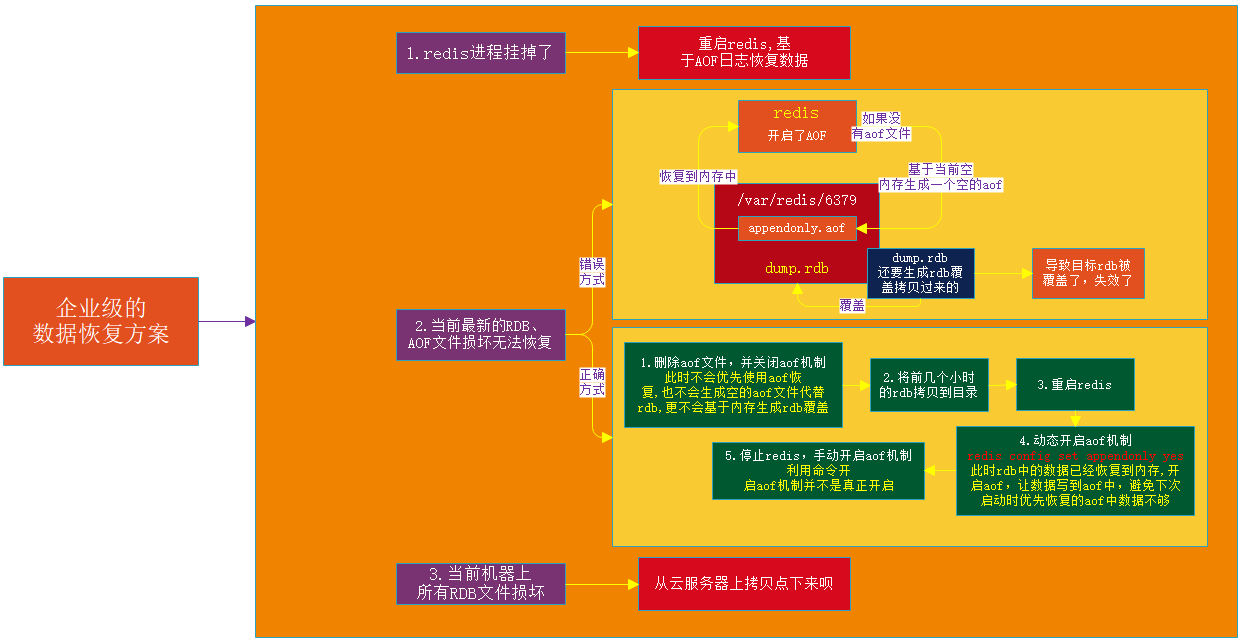


## 实战篇

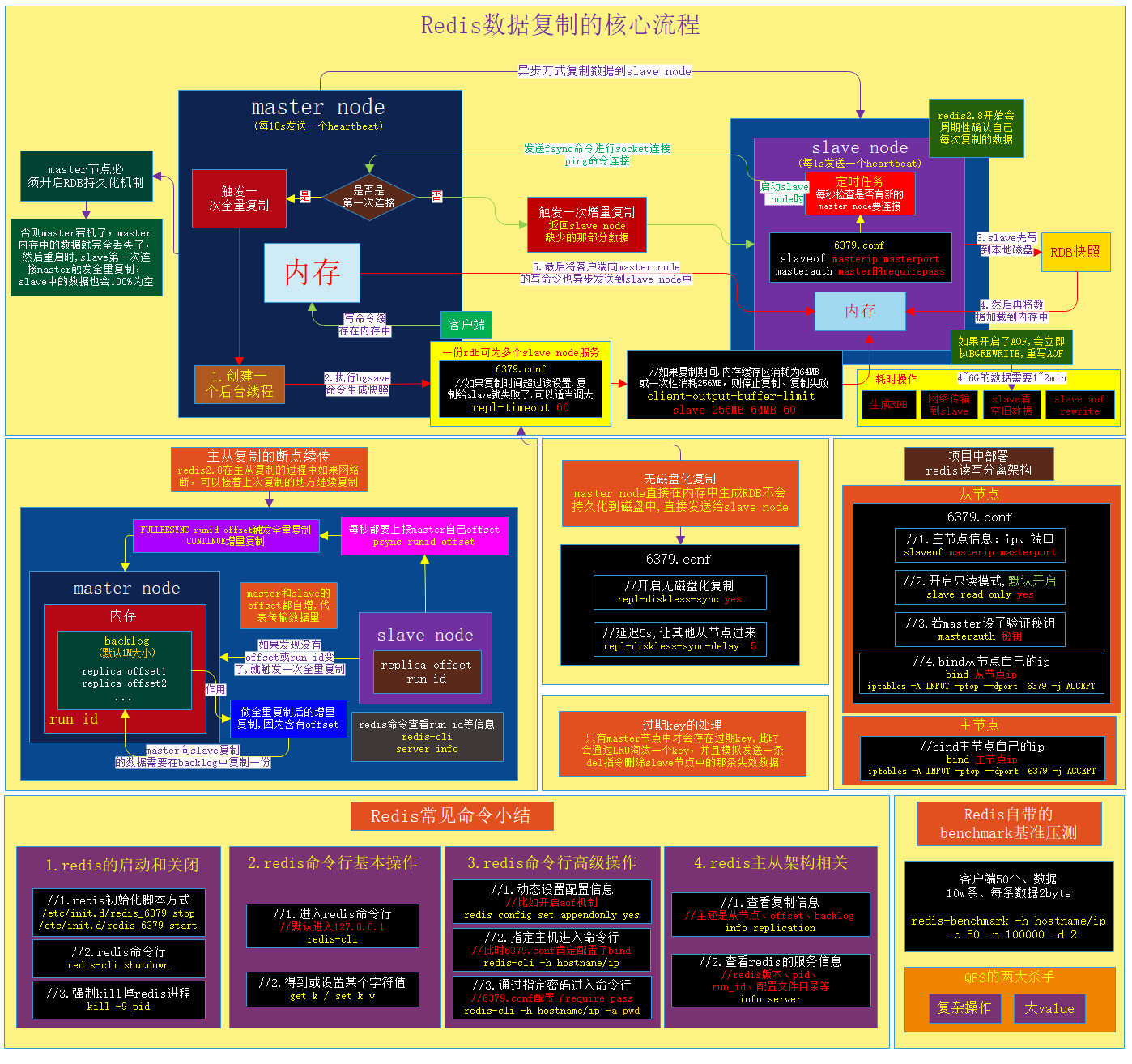


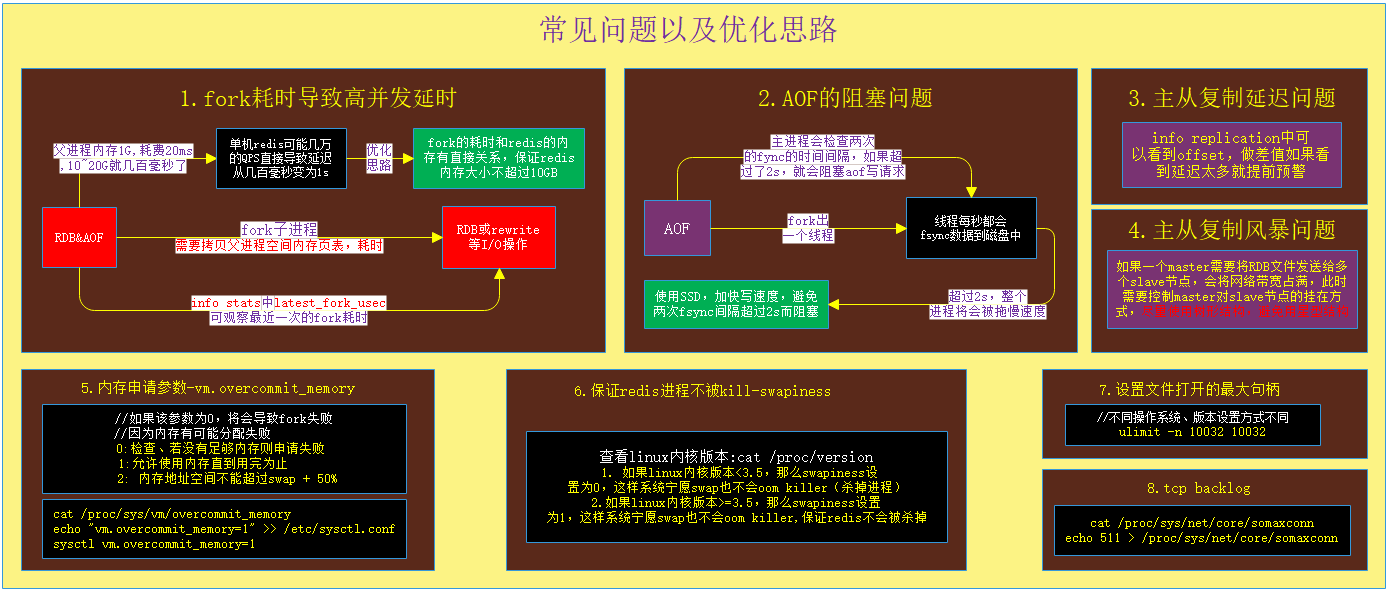






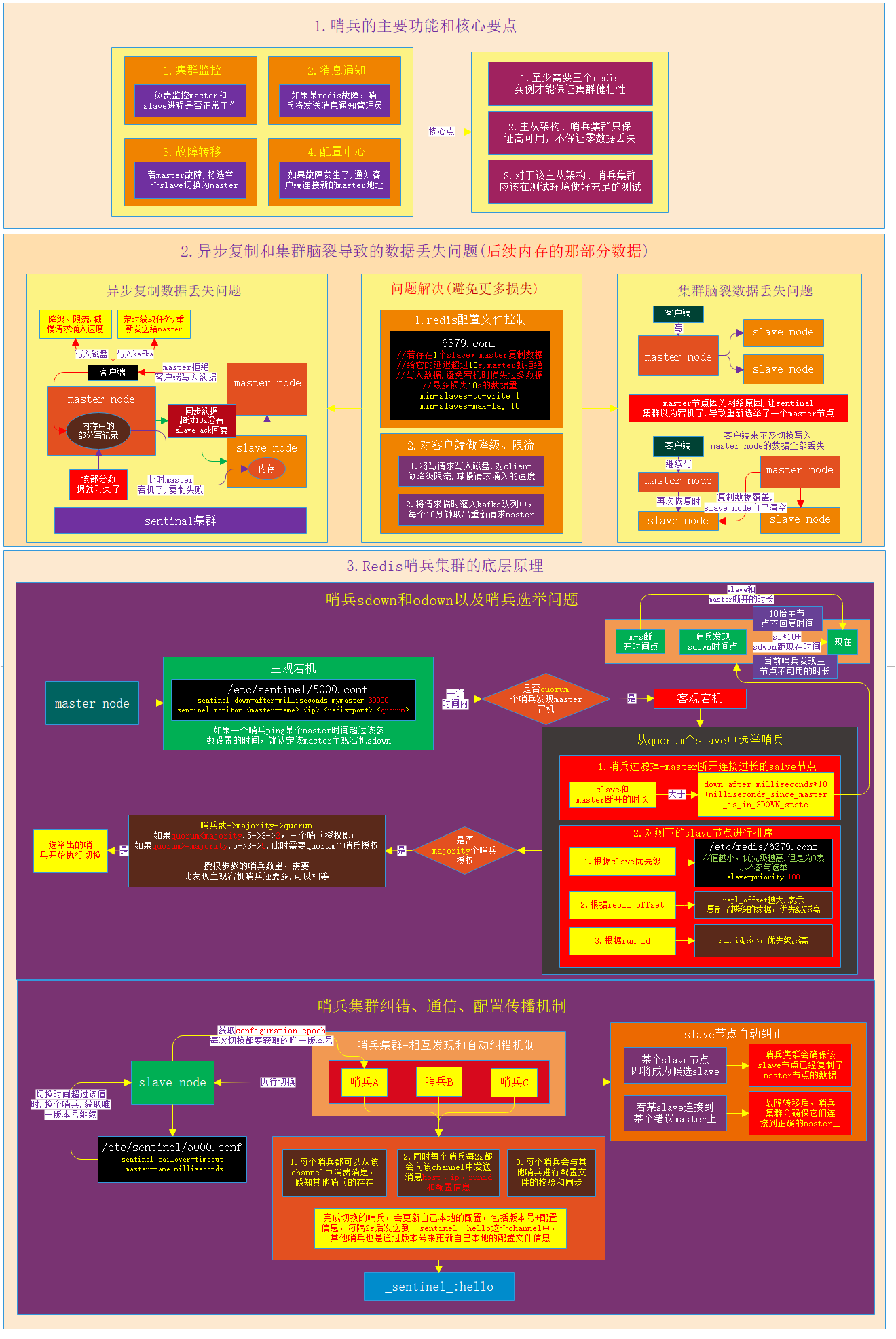
# 高并发：Redis主从架构-读写分离-支持水平扩容读高并发架构



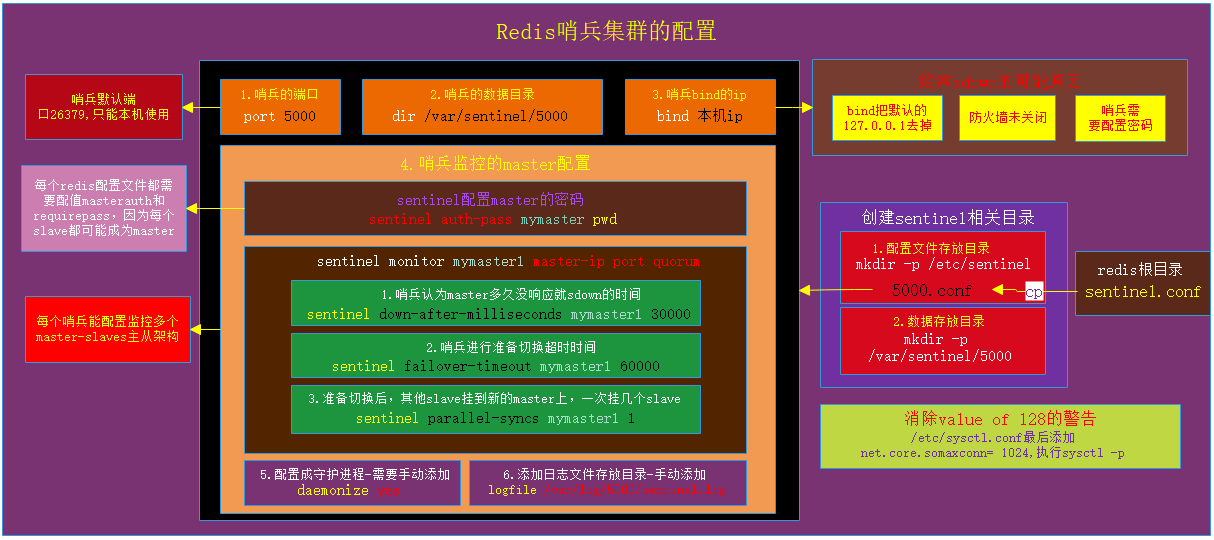


# 高可用：Redis主从架构-哨兵集群构建99.99%的高可用性架构

## 原理篇

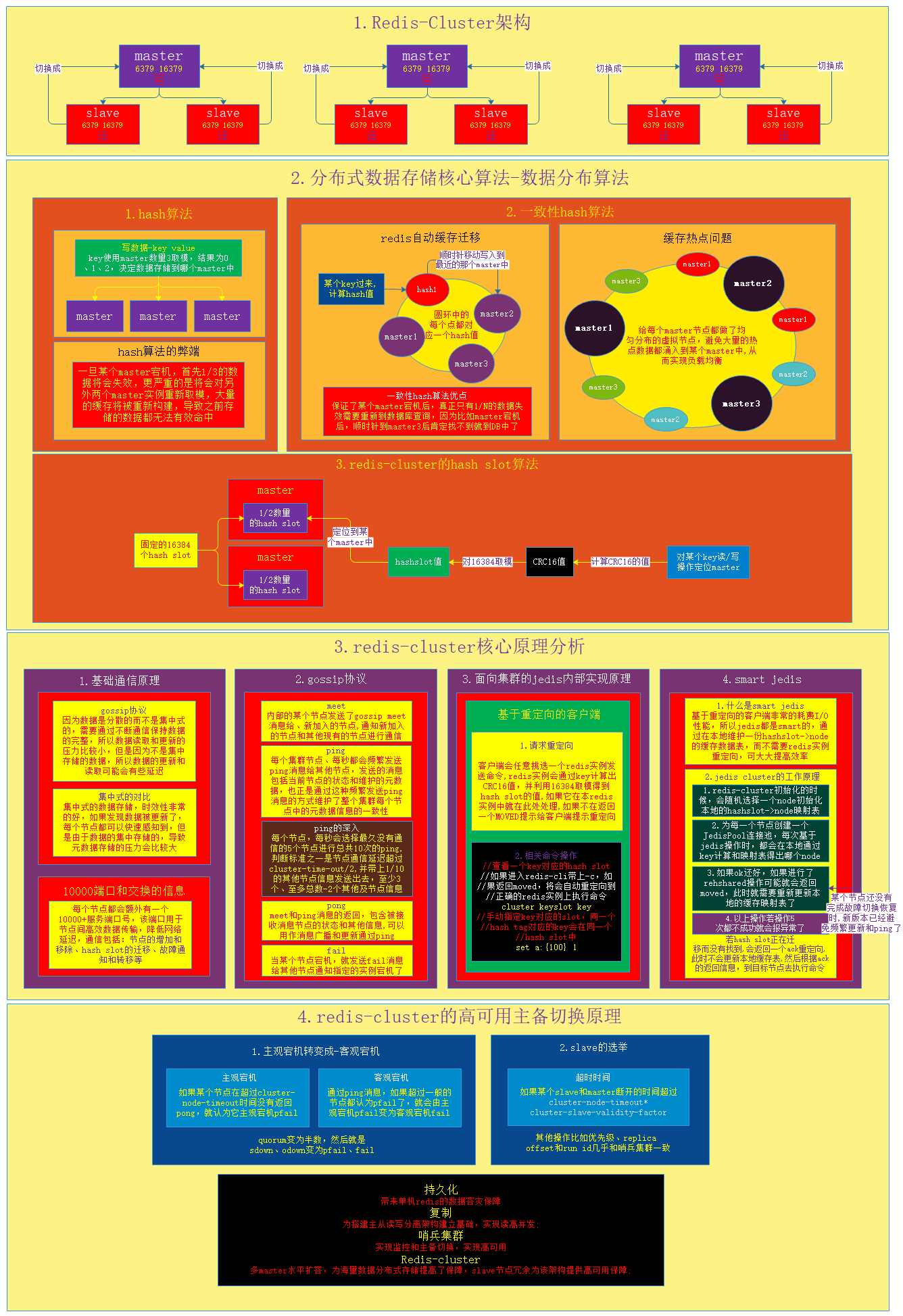


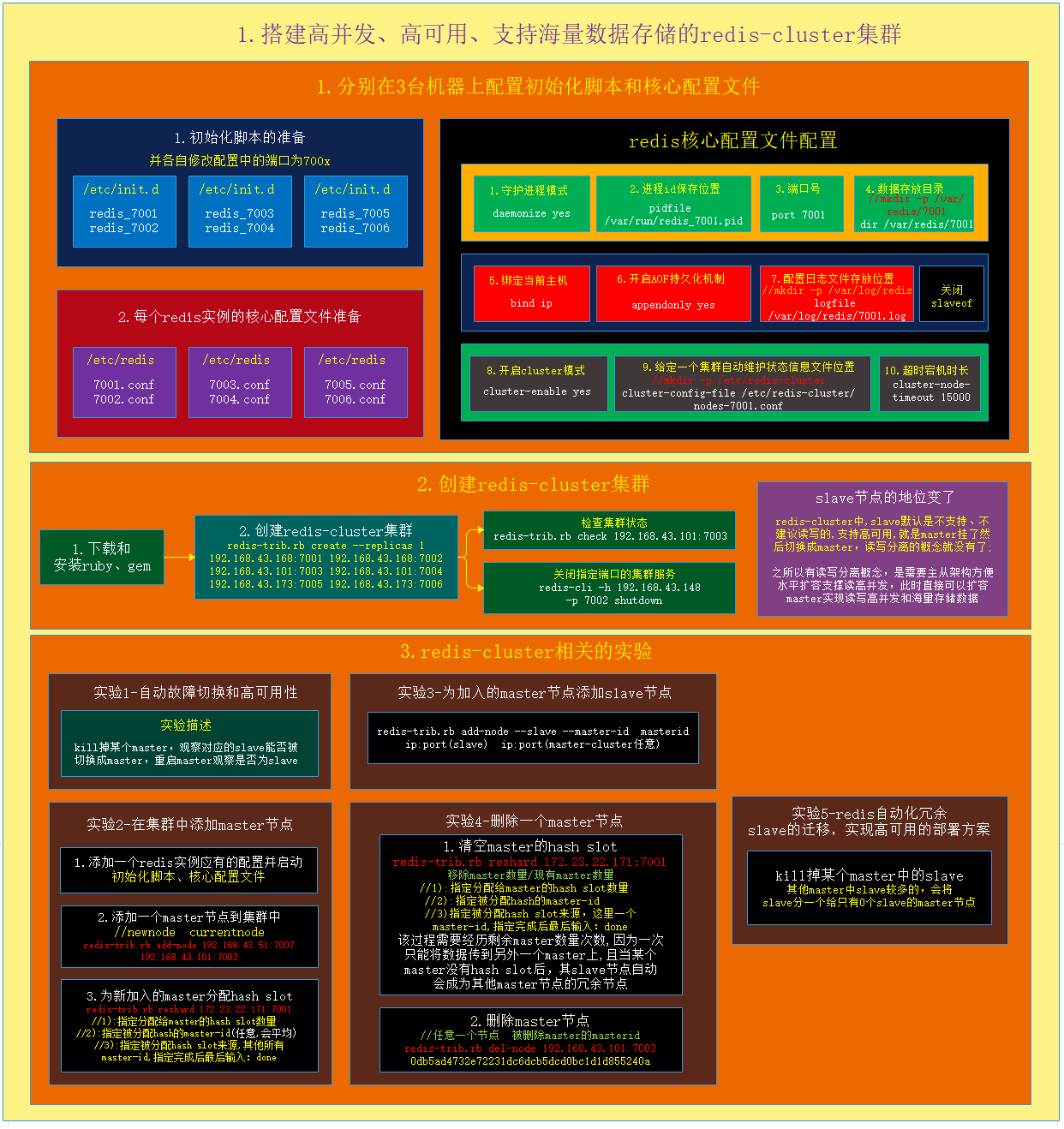
## 实战篇





# 海量数据：在Redis读写分离-高可用的架构下，构建支撑海量数据的架构





## 搭建redis-cluster集群架构

### 下载和安装ruby

wget https://cache.ruby-lang.org/pub/ruby/2.3/ruby-2.3.1.tar.gz

tar -zxvf ruby-2.3.1.tar.gz

cd ./ruby-2.3.1

./configure -prefix=/usr/local/ruby

make && make install

cd /usr/local/ruby

cp bin/ruby /usr/local/bin

cp bin/gem /usr/local/bin

/usr/local/bin/ruby --version

//因为系统找命令的文件是从/usr/bin中寻找的，所以需要再/usr/bin中创建//一个链接，指向/usr/local/bin;

ln -s /usr/local/bin/ruby /usr/bin/ruby

ln -s /usr/local/bin/gem /usr/bin/gem

### 下载和安装gem

wget http://rubygems.org/downloads/redis-3.3.0.gem

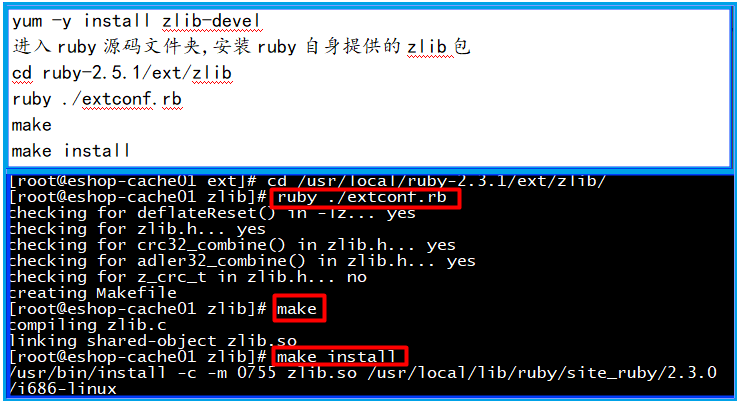
gem install -l ./redis-3.3.0.gem

gem --version

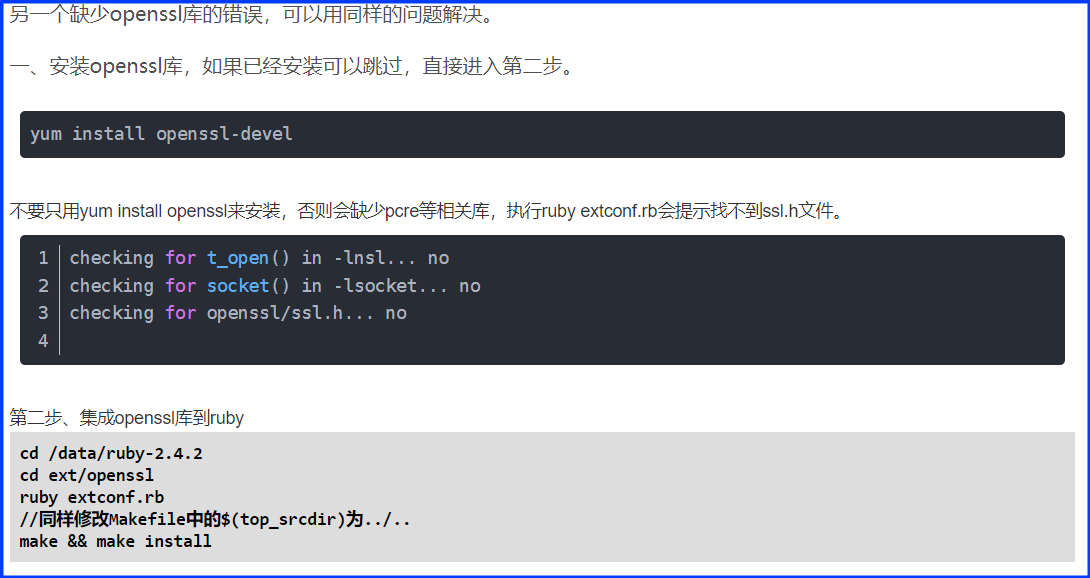
缺少zlib和没有安装openssl库



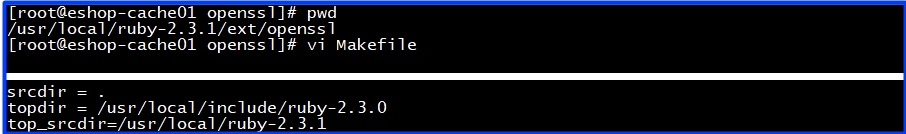
1.安装zlib



2.安装openssl







然后修改top\_srcdir为../..

top\_srcdir= /usr/local/ruby-2.3.1

srcdir = $(top\_srcdir)/ext/openssl

topdir = ../..

//当前目录下执行

make

make install

gem install -l ./redis-3.3.0.gem

gem --version

gem install redis

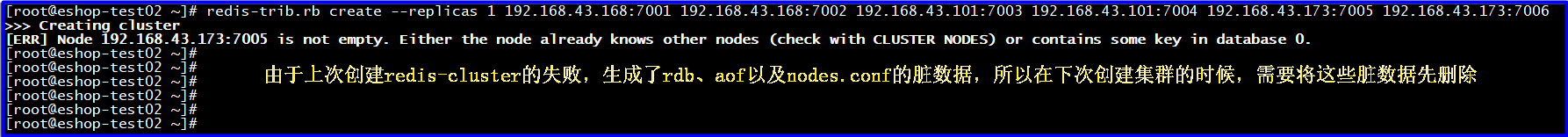
gem list --check redis gem

cp /usr/local/redis-3.2.8/src/redis-trib.rb /usr/local/bin

### 创建redis-cluster集群

//遇到提示一路yes加回车, --replicas: 每个master有几个slave

redis-trib.rb create --replicas 1 192.168.43.168:7001 192.168.43.168:7002 192.168.43.101:7003 192.168.43.101:7004 192.168.43.173:7005 192.168.43.173:7006



//检查集群状态

redis-trib.rb check 192.168.43.101:7003

//停止指定ip和端口上的redis服务

redis-cli -h 192.168.43.148 -p 7002 shutdown

## master水平扩容

### 在集群中添加一个master节点

1.准备master节点的信息

初始化脚本和配置文件，并启动master服务，准备加入集群中。

2.给新增master分配hash slot

连接到新的redis实例上,确认自己是否加入了集群作为了一个新的master;

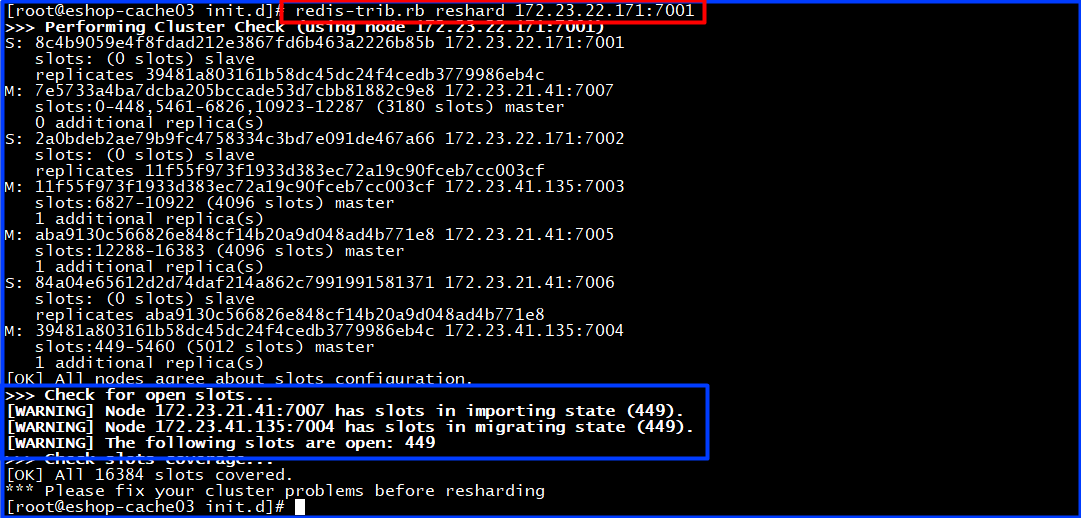
reshard一些数据过去,resharding的意思就是把一部分hash slot从一些node上迁移到另外一些node上

**redis-trib.rb reshard 172.23.22.171:7001**

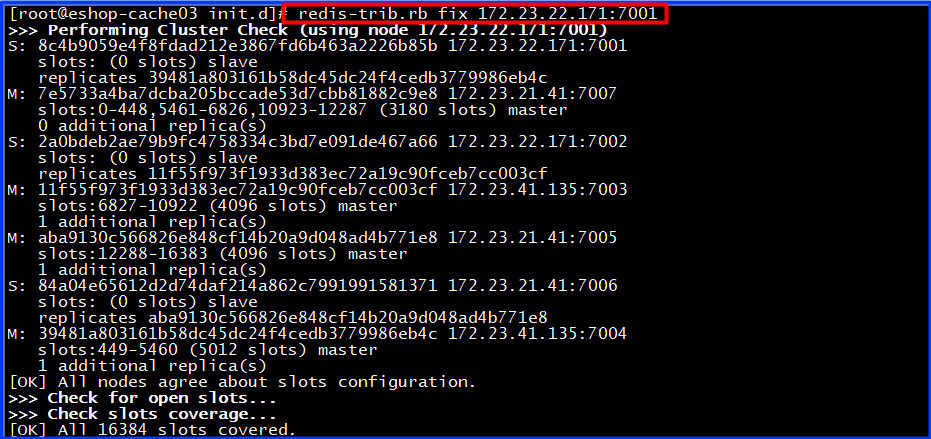
要把之前3个master上,总共4096个hashslot迁移到新的第四个master上去



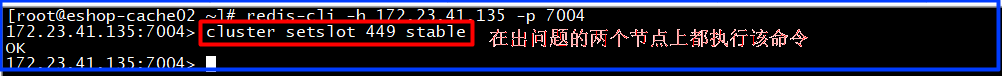
异常和解决方案

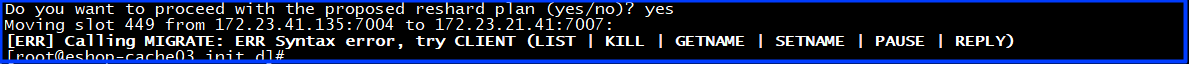


1.方法一：fix

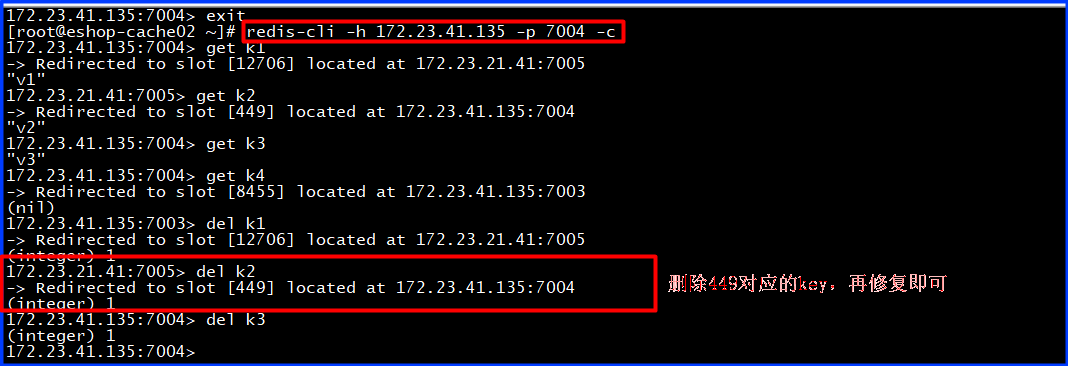


2.方法二：cluster setslot 449 stable





3.方法三：删除449 hash slot对应的key



### 为新加的master添加slave节点

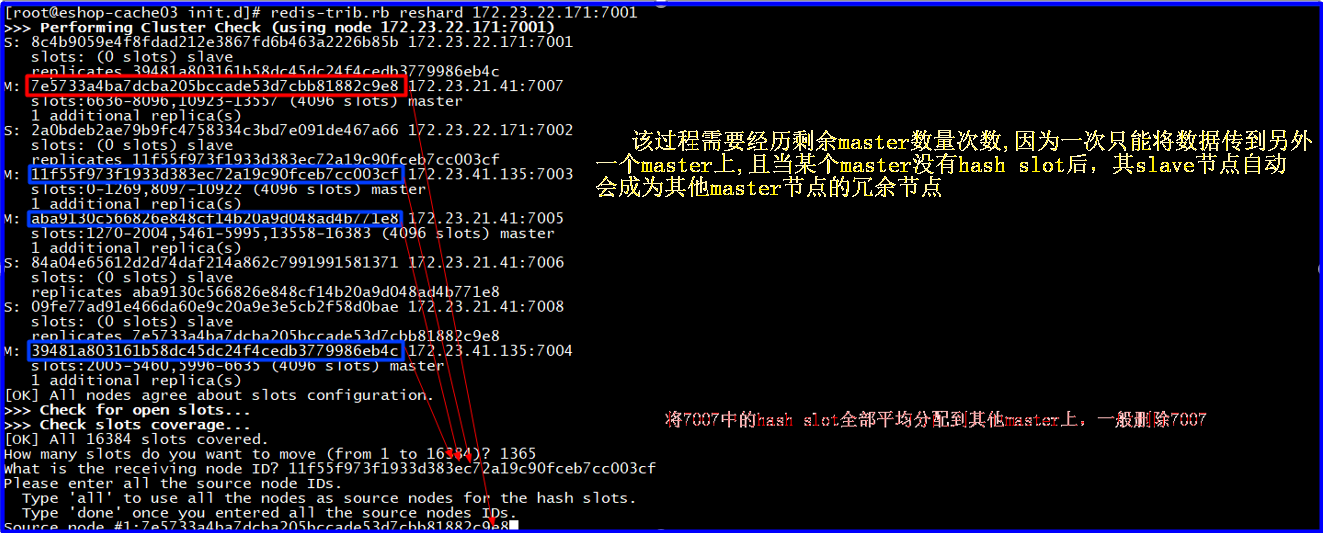
创建一个新的redis实例,再执行如下新增slave指令

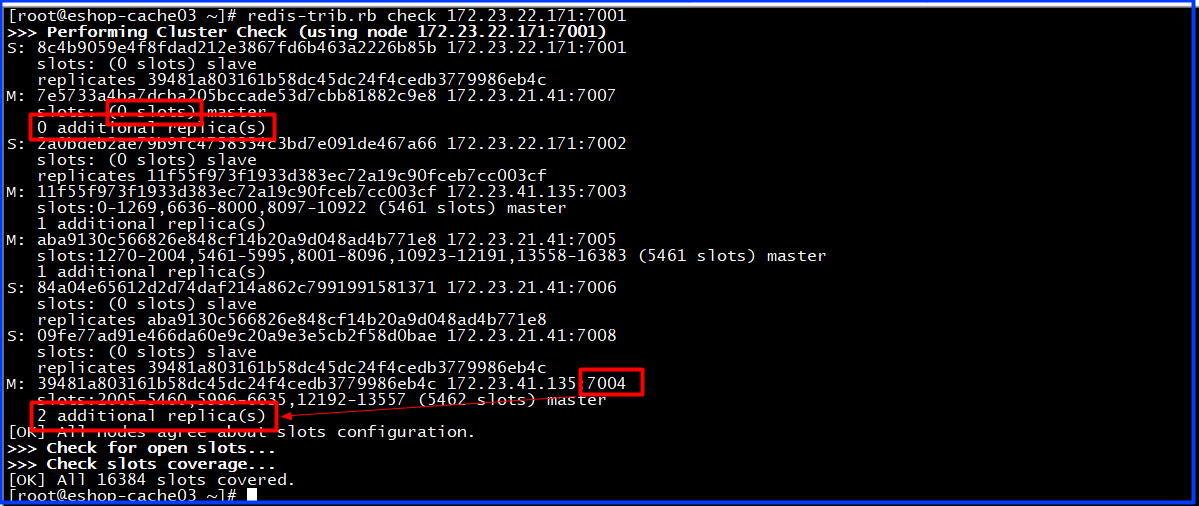
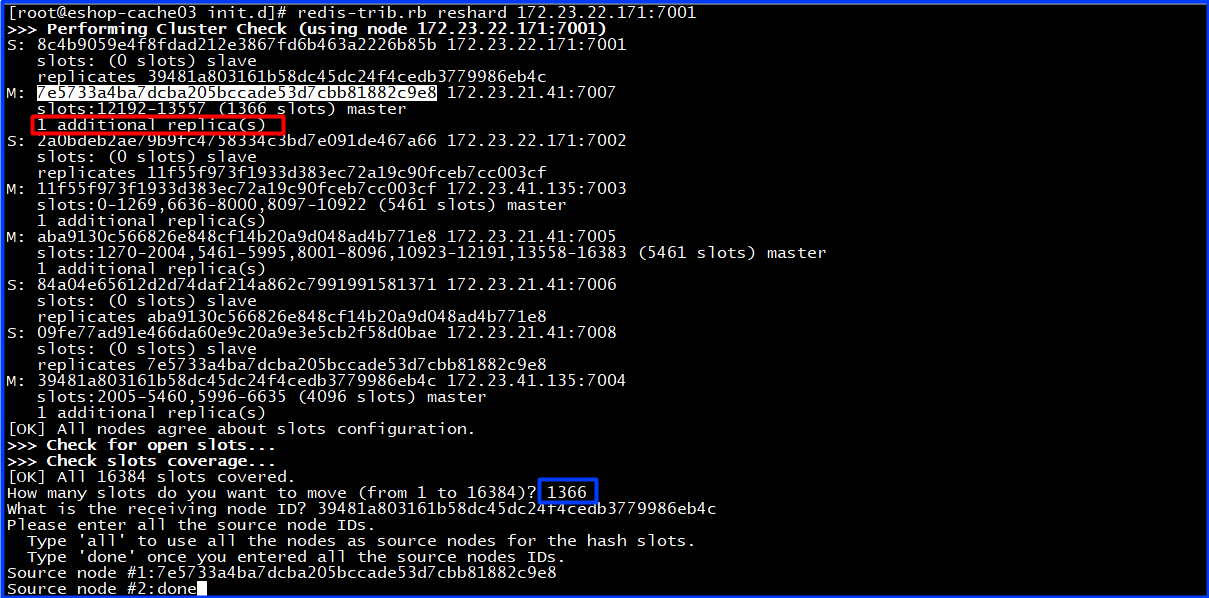
redis-trib.rb add-node --slave --master-id masterid ip:port(slave) ip:port(master-cluster任意)



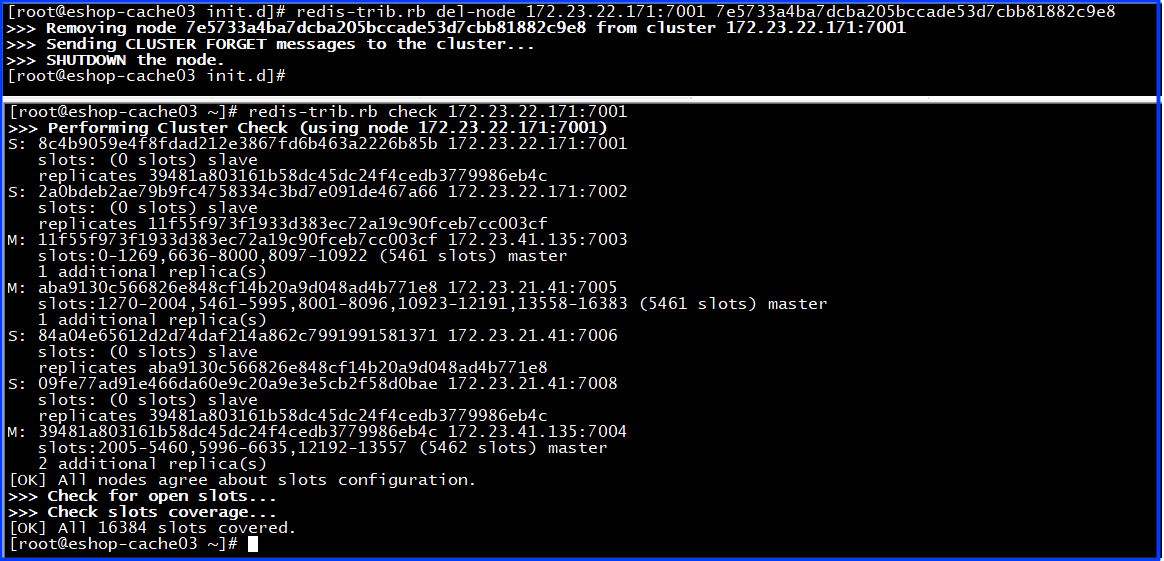
## 删除redis-cluster中的master

### 移除master中的hash slot

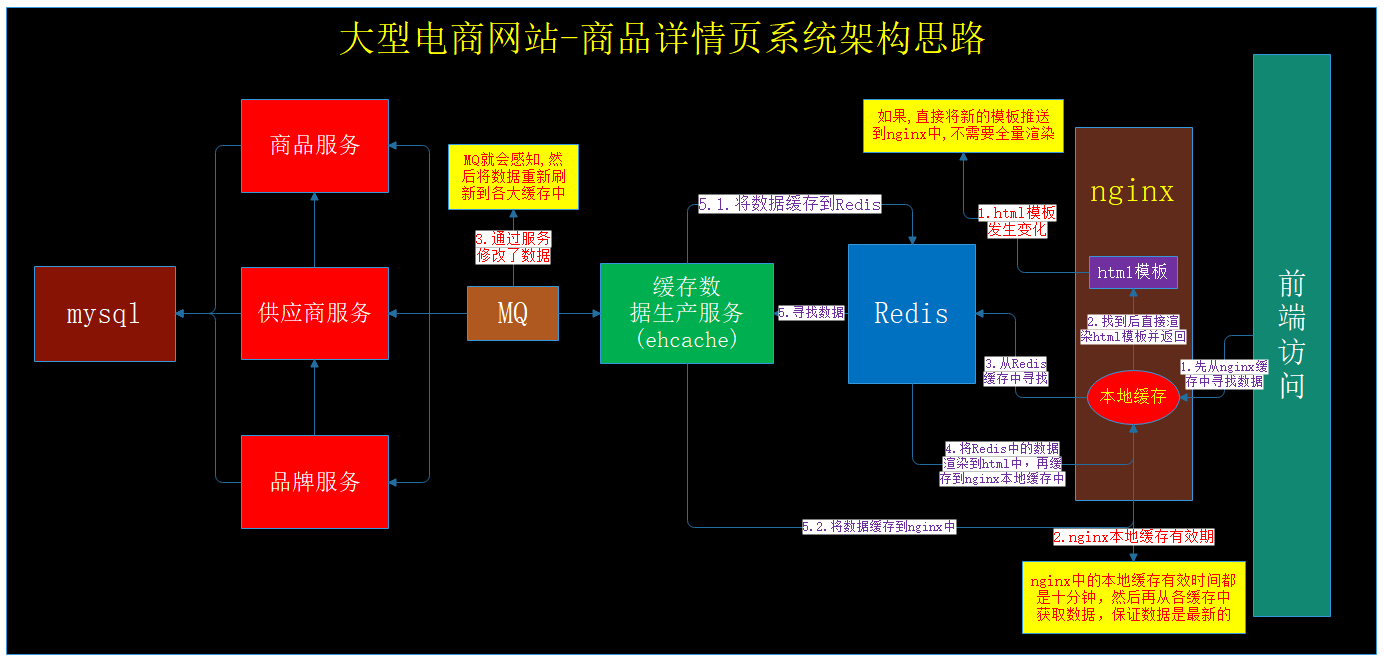




### 在集群中剔除master节点



# 大型电商网站-异步多级缓存构建-nginx数据本地化动态渲染的架构



# Cache Aside Pattern与数据库读写模式的分析

## 什么是cache aside pattern

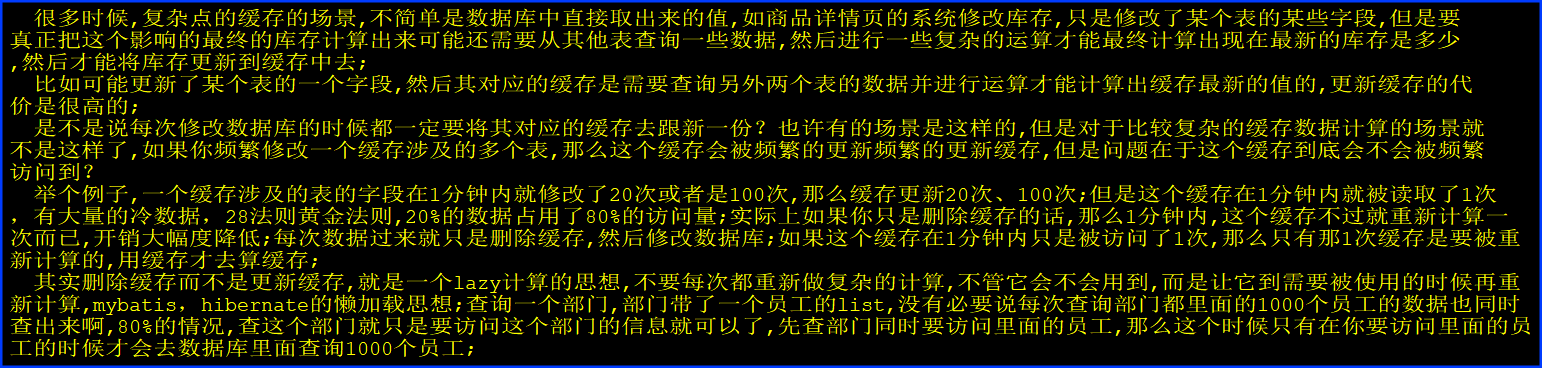
1.读时

先读缓存,缓存没有的话那么就读数据库,然后取出数据后放入缓存,同时返回响应;

2.更新时

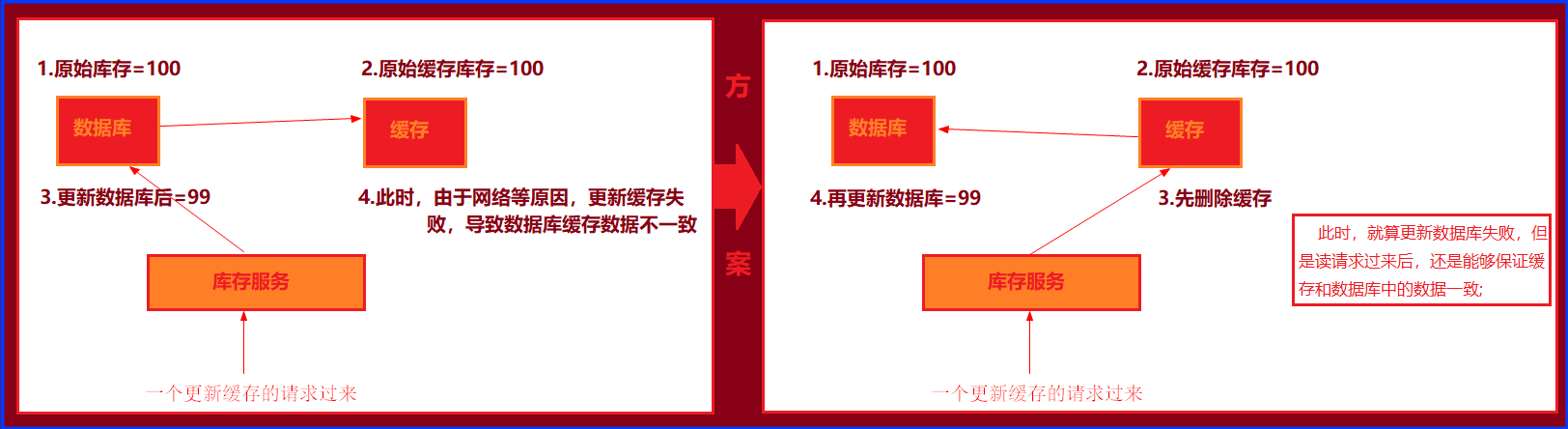
先删除缓存,然后再更新数据库;

## 更新的时候,为什么是删除缓存,而不是更新缓存呢？



# 高并发场景下的缓存、数据库双写不一致问题分析与解决方案设计

## 最初级的数据库缓存数据不一致问题与解决方案



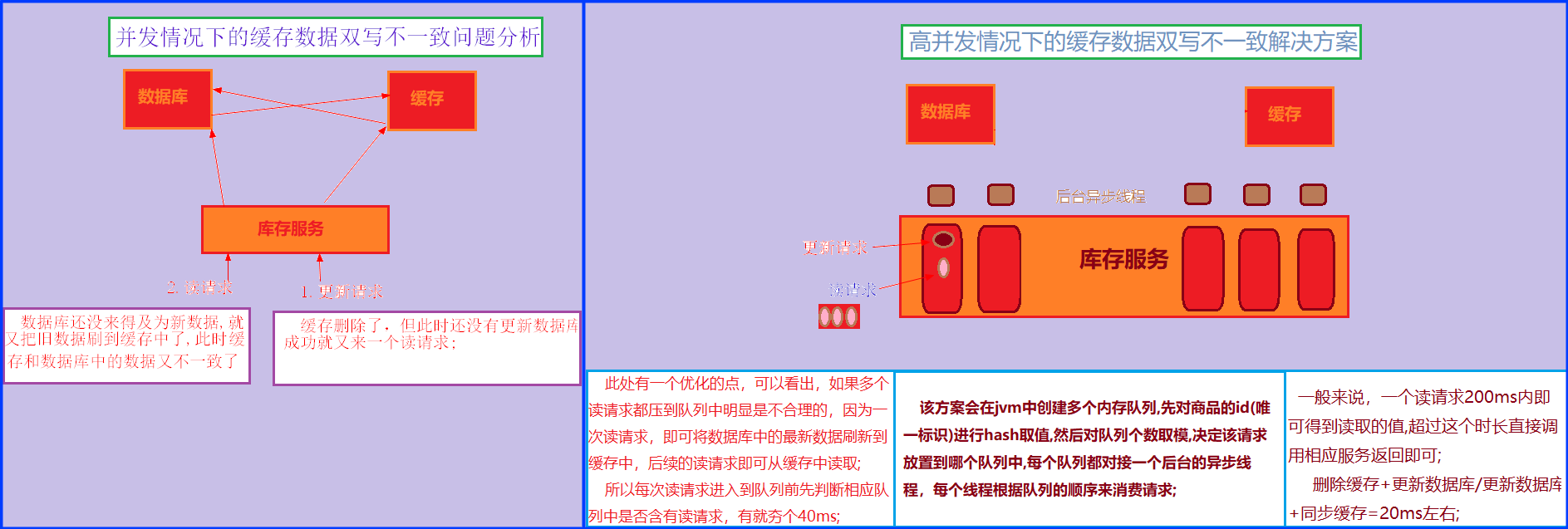
### 问题分析

先修改数据库再删除缓存,此时如果删除缓存失败了,那么会导致数据库中是新数据,缓存中是旧数据,数据缓存出现数据不一致;

### 解决方案

换个顺序,先删除缓存再修改数据库,如果删除缓存成功了而修改数据库失败了,那么数据库中是旧数据,缓存中是空的,那么数据不会不一致,因为读的时候缓存没有,则读数据库中旧数据会更新到缓存中;

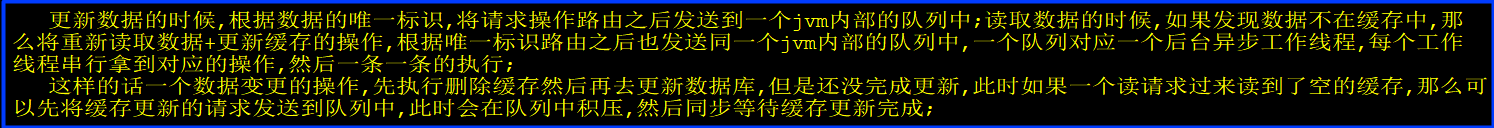
## 比较复杂的数据库缓存数据不一致问题分析



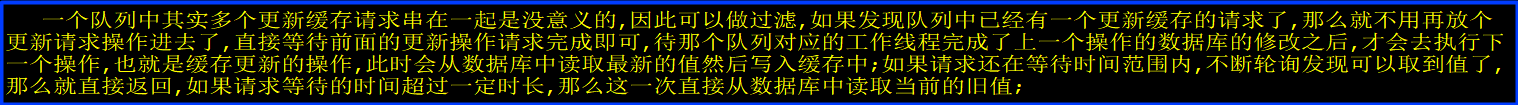
### 问题分析

沿用初级问题的解决方案,此时数据发生了变更,先删除了缓存,然后要去修改数据库,此时还没修改数据库成功;一个读请求过来去读缓存发现缓存空了,去查询数据库,查到了修改前的旧数据并放到了缓存中,而第一个数据变更的变更程序完成了数据库的修改,此时数据库和缓存中的数据又不一样了;

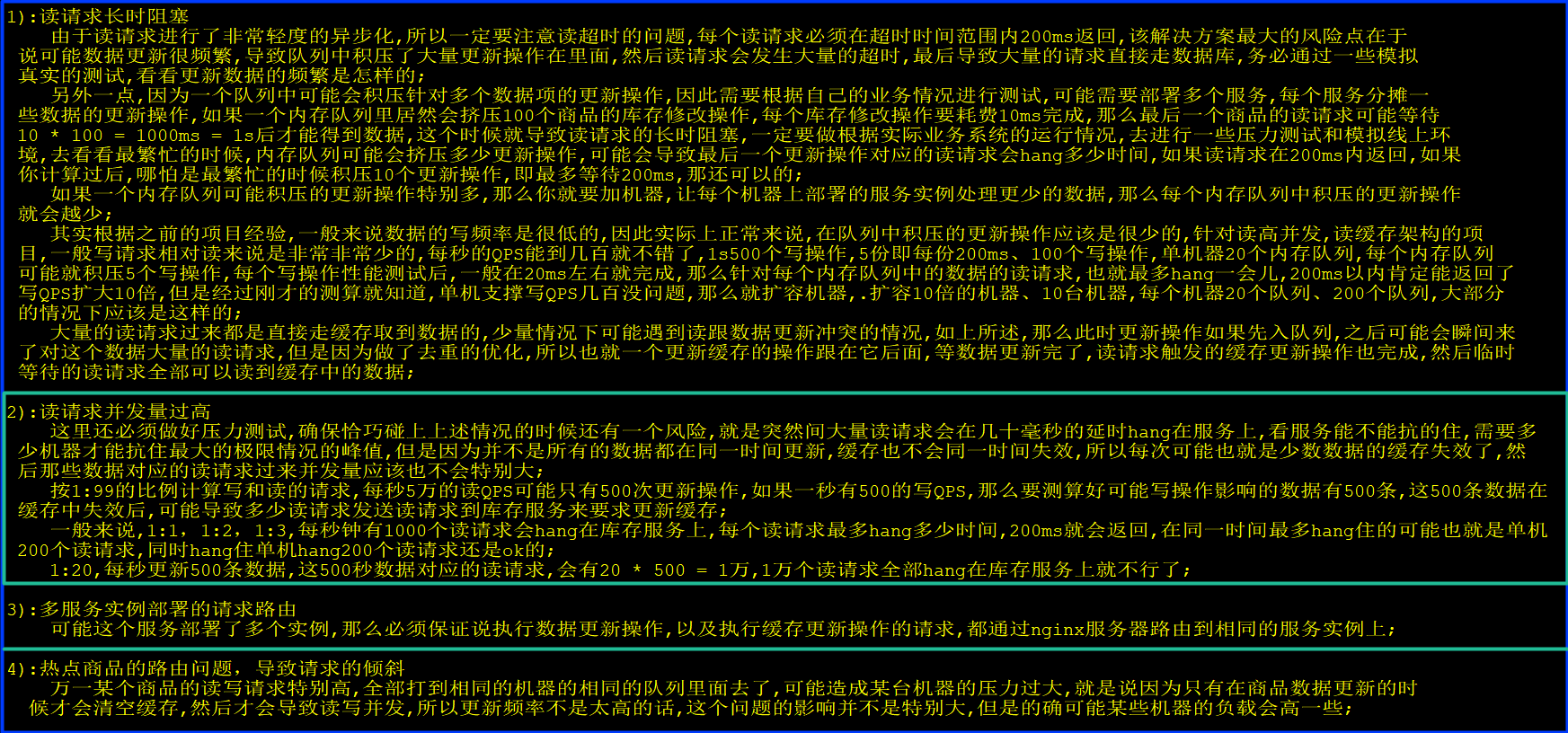
### 解决方案-数据库与缓存更新与读取操作进行异步串行化



### 方案优化点



### 高并发的场景下,该解决方案要注意的问题



# Eshop-库存项目的初始化工作

## 通过yum安装mysql数据库的

yum install -y mysql-server//安装数据库5.1.73

service mysqld start//启动mysql服务

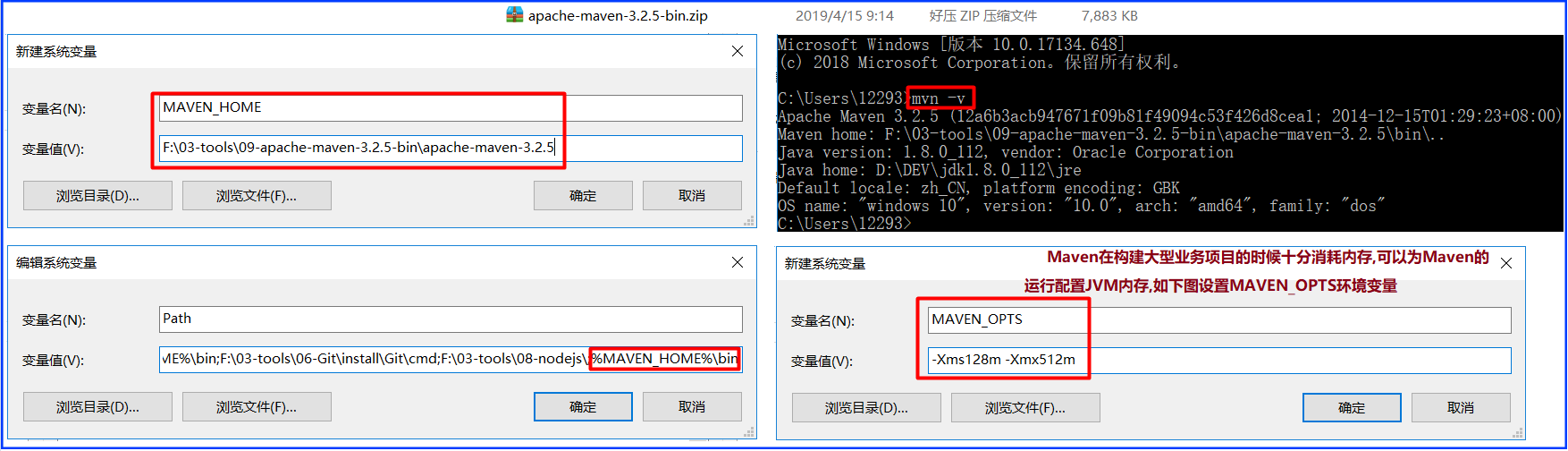
chkconfig mysqld on//设置开机时自动启动

yum install -y mysql-connector-java//安装mysql客户端

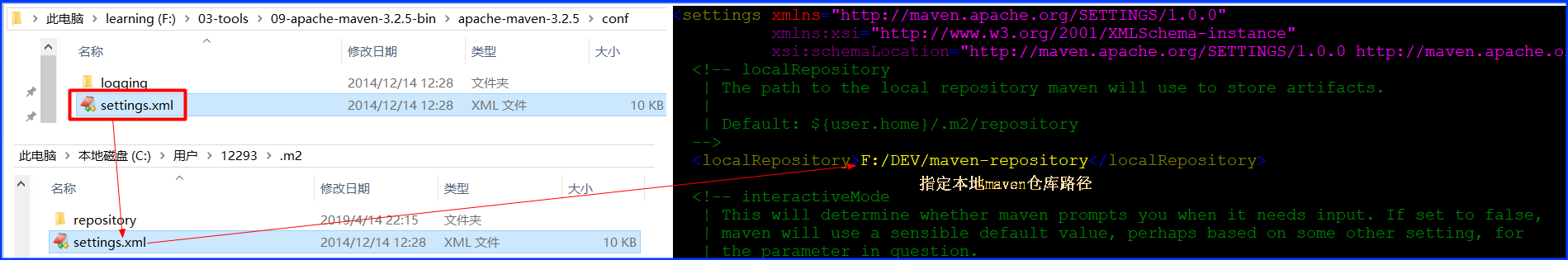


## Maven的安装和配置

### 下载并配置maven的环境变量



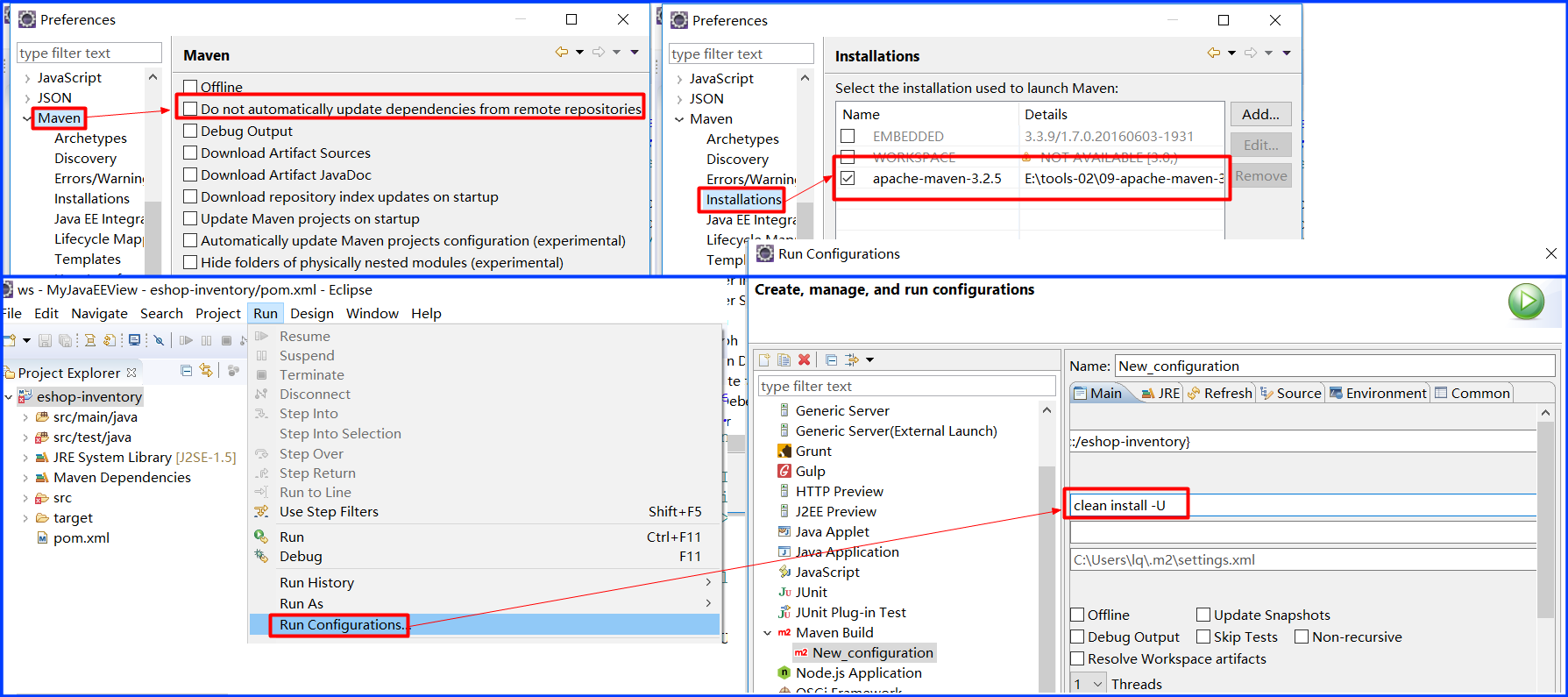
### 设置maven的本地仓库



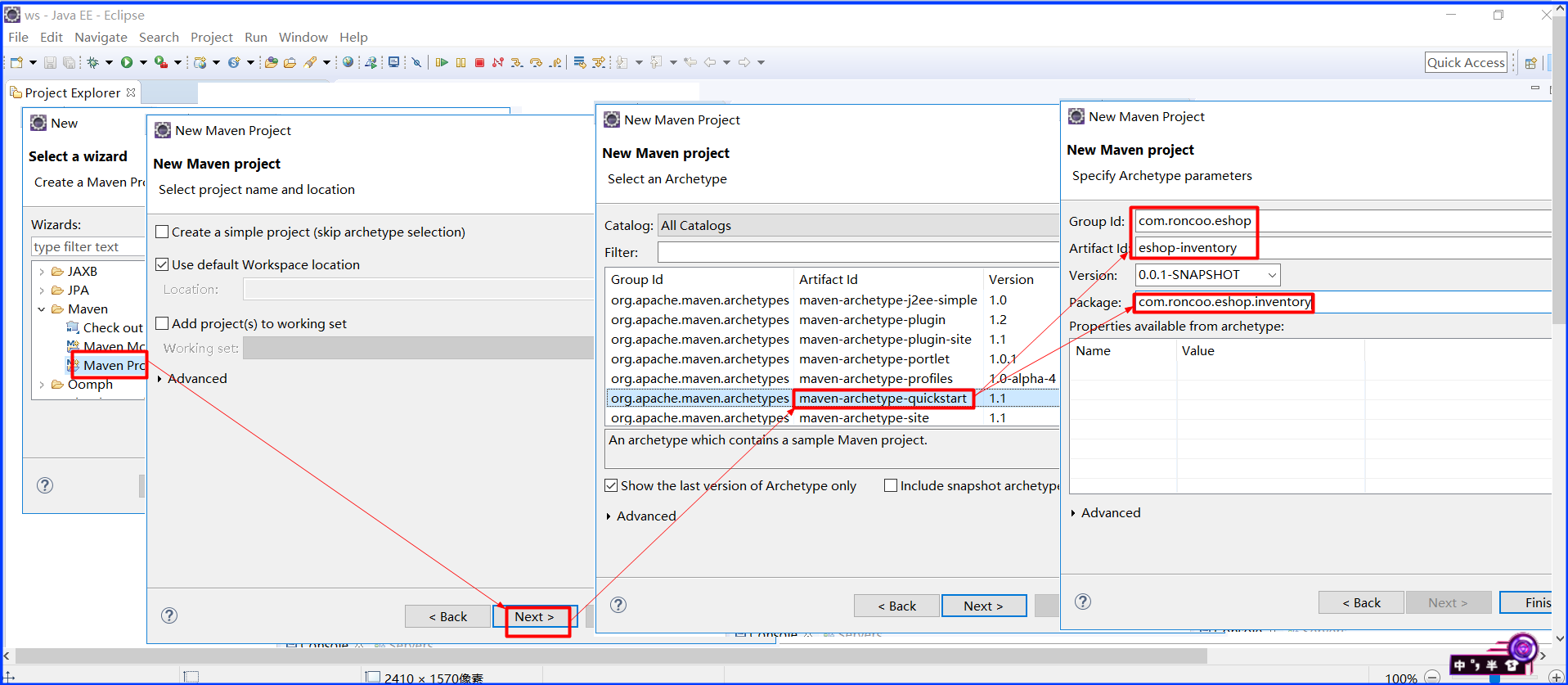
### Eclipse中关联maven



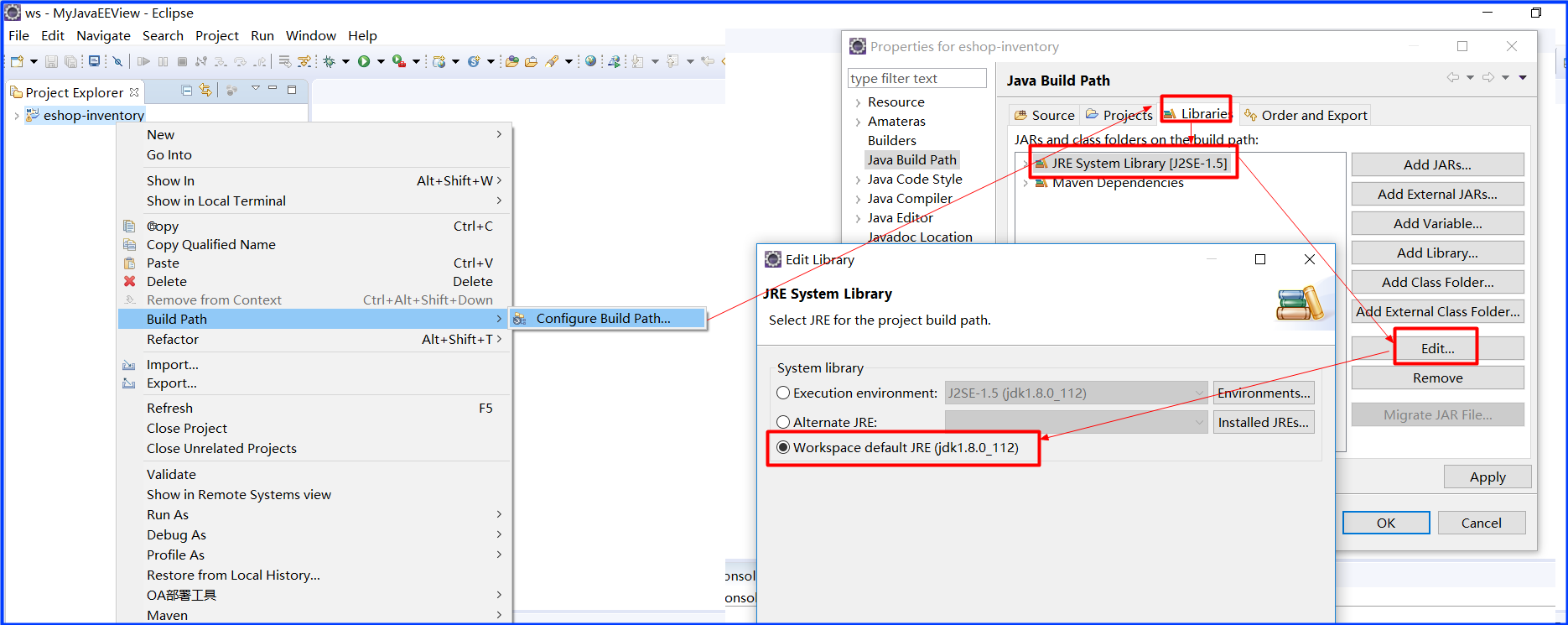
### Maven配置Eclipse自动下载



## Eshop-库存项目的maven项目创建



1. jdk修改为默认的1.8



1. j