【Linux基础篇】

1.描述**Linux**运行级别**0-6**的各自含义

```
0：关机模式
1：单用户模式<==破解root密码
2：无网络支持的多用户模式
3：有网络支持的多用户模式（文本模式，工作中最常用的模式）
4：保留，未使用
5：有网络支持的X-windows支持多用户模式（桌面）
6：重新引导系统，即重启
```

2.描述**Linux**系统从开机到登陆界面的启动过程

```
1.开机BIOS自检，加载硬盘。
2.读取MBR,MBR引导。
3.grub引导菜单(Boot Loader)。
4.加载内核kernel。
5.启动init进程，依据inittab文件设定运行级别
6.init进程，执行rc.sysinit文件。
7.启动内核模块，执行不同级别的脚本程序。
8.执行/etc/rc.d/rc.local
9.启动mingetty，进入系统登陆界面。
```

3.描述**Linux**下软链接和硬链接的区别

```
在Linux系统中，链接分为两种，一种是硬链接（Hard link），另一种称为符号链接或软链接（Symbolic Link）。
1.默认不带参数的情况下，ln创建的是硬链接，带-s参数的ln命令创建的是软链接。
2.硬链接文件与源文件的inode节点号相同，而软链接文件的inode节点号，与源文件不同，
3.ln命令不能对目录创建硬链接，但可以创建软链接。对目录的软链接会经常使用到。
4.删除软链接文件，对源文件和硬链接文件无任何影响。
5.删除文件的硬链接文件，对源文件及软链接文件无任何影响。
6.删除链接文件的源文件，对硬链接文件无影响，会导致其软链接失效（红底白字闪烁状）。
7.同时删除源文件及其硬链接文件，整个文件才会被真正的删除。
8.很多硬件设备的快照功能，使用的就是类似硬链接的原理。
9.软链接可以跨文件系统，硬链接不可以跨文件系统。
```

4.如果一台办公室内主机无法上网（打不开网站），请给出你的排查步骤？

```
1.首先确定物理链路是否联通正常。
2.查看本机IP，路由，DNS的设置情况是否达标。
3.telnet检查服务器的WEB有没有开启以及防火墙是否阻拦。
4.ping一下网关，进行最基础的检查，通了，表示能够到达服务器。
5.测试到网关或路由器的通常情况，先测网关，然后再测路由器一级一级的测试。
6.测试ping公网ip的通常情况（记住几个外部IP），
7.测试DNS的通畅。ping出对应IP。
8.通过以上检查后，还在网管的路由器上进行检查。
```

5.网站打开慢，请给出排查方法，如是数据库慢导致，如何排查并解决，请分析并举例？

1、可以使用top free 等命令分析系统性能等方面的问题
2、如是因为数据库的原因造成的，就需要查看慢查询日志去查找并分析问题所在

6.如何选择Linux操作系统版本?

一般来讲，桌面用户首选Ubuntu；服务器首选RHEL或CentOS，两者中首选CentOS。
根据具体要求：
1.安全性要求较高，则选择Debian或者FreeBSD。
2.需要使用数据库高级服务和电子邮件网络应用的用户可以选择SUSE。
3.想要新技术新功能功能可以选择Feddora，Feddora是RHEL和CentOS的一个测试版和预发布版本。
4.根据现有状况，绝大多数互联网公司选择CentOS。现在比较常用的是6系列，现在市场占有大概一半左右。另外的原因是CentOS更侧重服　务器领域，并且无版权约束。

7.生产场景如何对linux系统进行合理规划分区？

分区的根本原则是简单、易用、方便批量管理。根据服务器角色定位建议如下：
1.单机服务器：如8G内存，300G硬盘
　分区：/boot 100-200M，swap 16G，内存大小8G*2，/ 80G，/var 20G（也可不分），/data 180G（存放web及db数据）
　优点：数据盘和系统盘分开，有利于出问题时维护。
　RAID方案：视数据及性能要求，一般可采用raid5折中。

2.负载均衡器（如LVS等）
　分区：/boot 100-200M，swap 内存的1-2倍，/　，
　优点：简单方便，只做转发数据量很少。
　RAID方案：数据量小，重要性高，可采用RAID1

3.负载均衡下的RS server
　分区：/boot 100-200M，swap 内存的1-2倍，/
　优点：简单方便，因为有多机，对数据要求低。
　RAID方案：数据量大，重要性不高，有性能要求，数据要求低，可采用RAID0

4.数据库服务器mysql及oracle如16/32G内存
　分区：/boot 100-200M，swap 16G，内存的1倍，/ 100G，/data 剩余（存放db数据）
　优点：数据盘和系统盘分开，有利于出问题时维护,及保持数据完整。
　RAID方案：视数据及性能要求主库可采取raid10/raid5，从库可采用raid0提高性能（读写分离的情况下。）

5.存储服务器
　分区：/boot 100-200M，swap 内存的1-2倍，/ 100G，/data(存放数据)
　优点：此服务器不要分区太多。只做备份，性能要求低。容量要大。
　RAID方案：可采取sata盘，raid5

6.共享存储服务器（如NFS）
　分区：/boot 100-200M，swap 内存的1-2倍，/ 100G，/data(存放数据)
　优点：此服务器不要分区太多。NFS共享比存储多的要求就是性能要求。
　RAID方案：视性能及访问要求可以raid5,raid10,甚至raid0（要有高可用或双写方案）

7.监控服务器cacti,nagios
　分区：/boot 100-200M，swap 内存的1-2倍，/
　优点：重要性一般，数据要求也一般。
　RAID方案：单盘或双盘raid1即可。三盘就RAID5，看容量要求加盘即可。

8.如何查看当前的**Linux**服务器的运行级别？

```
‘who -r’ 和 ‘runlevel’命令可以用来查看当前的Linux服务器的运行级别。
```

9.请简述如何查看**Linux**的系统版本

```
uname -a
```

10.查看**Linux**运行多少时间

```
uptime
```

11.某系统管理员需每天做一定的重复工作，请按照下列要求，编制一个解决方案：

```
（1）在下午4:50删除/abc目录下的全部子目录和全部文件；
（2）从早8:00～下午6:00每小时读取/xyz目录下x1文件中每行第一个域的全部数据加入到/backup目录下的bak01.txt
文件内；
（3）每逢星期一下午5:50将/data目录下的所有目录和文件归档并压缩为文件：backup.tar.gz；

(1)  crontab -e   50 16 * * * rm -rf /abc/

(2)  * 8-18/1 * * *  awk '{print $1 > "/backup/bak01.txt"}' /xyz/x1

(3)  50 17 * * 1 tar -czf backup.tar.gz /data
```

12.请在**3**月**23**号**21**点**18**分的时候，重启服务器

```
crontab -e 18 21 23 3 * init 6
```

13.如何查看**Linux**的默认网关？

```
用 “route -n” 和 “netstat -nr” 命令，我们可以查看默认网关。
除了默认的网关信息，这两个命令还可以显示当前的路由表。
```

14.如何查看一个文件夹**inode**节点数有多少？

```
find / -xdev -printf '%h\n' | sort | uniq -c | sort -k 1 -n
```

15.写一个脚本查找最后创建时间是**3**天前，后缀是***.log**的文件并删除

```
find / -name “*.log” -ctime +3 -exec rm -f {} \;
```

16.如果某文件夹下文件太多无法**ls**该如何解决？

```
ls -f
```

17.如何用**tcpdump**嗅探**80**端口的访问看看谁最高？

```
tcpdump -i eth0 -tnn dst port 80 -c 1000 | awk -F”.” ‘{print $1”.”$2”.”$3”.”$4”.”}’ | sort |uniq -c | sort -nr | head-5
```

18.如何查看**/var/log**目录下的文件数？

```
ls /var/log/ -1R | grep “-” | wc -l
```

19.如何查看**Linux**系统每个**ip**的连接数？

```
netstat -n | awk ‘/^tcp/ {print $5}’ | awk -F: ‘{print $1}’ | sort | uniq -c | sort -rn
```

20.shell下生成**32**位随机密码

```
cat /dev/urandom | head -1 | md5sum | head -c 32 >> /pass
```

21.统计出**apache**的**access.log**中访问量最多的**5**个**ip**

```
cat access.log | awk ‘{print $1}’ | sort | uniq -c | sort -n -r | head -5
```

22.请用多种方式在**linux**系统中设置环境变量，并指出各种方式的区别。

```
1、控制台中设置，不赞成这种方式，因为他只对当前的shell 起作用，换一个shell设置就无效了：
PATH="PATH":/NEW_PATH (关闭shell Path会还原为原来的path)
2、修改 /etc/profile 文件，如果你的计算机仅仅作为开发使用时推存使用这种方法，因为所有用户的shell都有权使用这个环境变量，可能会给系统带来安全性问题。这里是针对所有的用户的，所有的shell 在/etc/profile的最下面添加： export PATH="$PATH:/NEW_PATH"
3、修改bashrc文件，这种方法更为安全，它可以把使用这些环境变量的权限控制到用户级别，这里是针对某一特定的用户，如果你需要给某个用户权限使用这些环境变量，你只需要修改其个人用户主目录下的 .bashrc文件就可以了。 在下面添加： Export PATH="$PATH:/NEW_PATH"
```

【基础服务篇】

23.写一个防火墙配置脚本，只允许远程主机访问本机的**80**端口。

```
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -F
iptables -X
iptables -A INPUT -i eth0 -p tcp –dport 80 -j ACCEPT
iptables -P INPUT DROP
```

24.如何将本地**80**端口的请求转发到**8080**端口，当前主机**IP**为**192.168.2.1**

```
/sbin/iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT--to 192.168.2.1:8080

/sbin/iptables -t nat -A PREROUTING -p tcp--dport 80 -j REDIRECT --to 8080
```

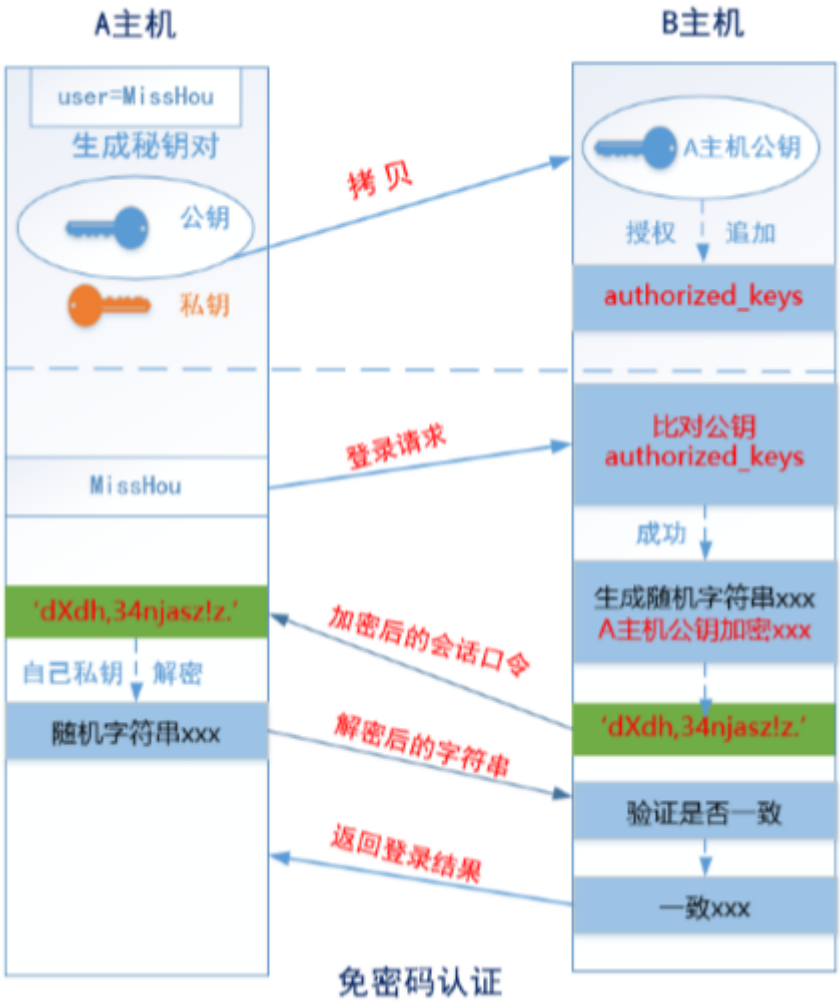
25.说出**5**个以上常用的服务端口

```
21------ftp                22------ssh                23------telnet
25------snmp               110-----ppop3              143-----IMAP
873-----rsync              80------http               3306------mysql
```

26.FTP的主动模式和被动模式

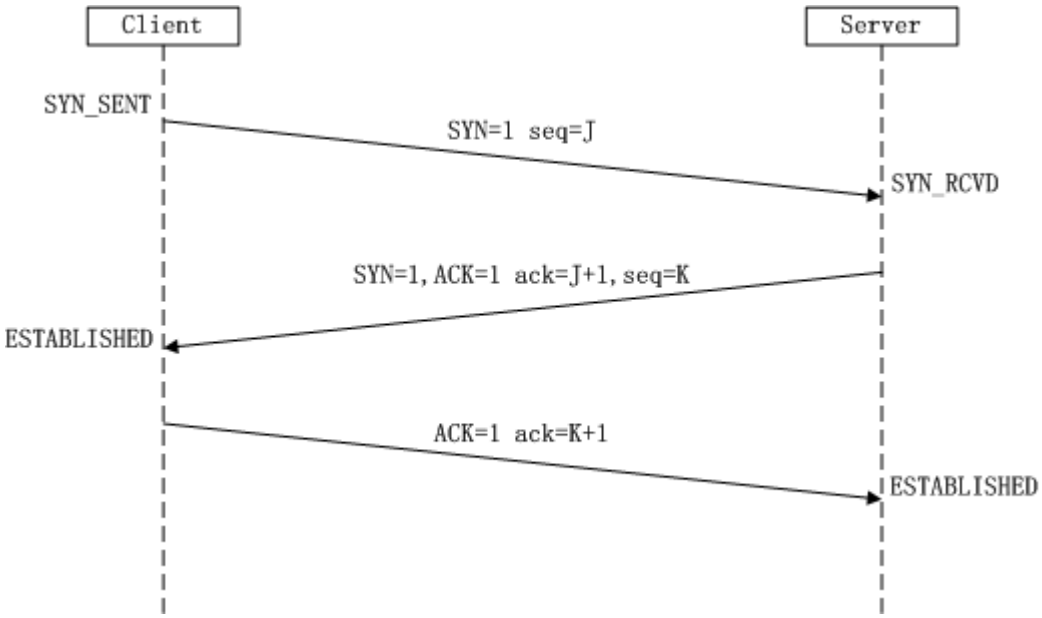
```
FTP协议有两种工作方式：PORT方式和PASV方式，中文意思为主动式和被动式。

- 主动模式

1. 客户端打开大于1023的随机命令端口和大于1023的随机数据端口向服务的的21号端口发起请求
2. ==服务端==的21号命令端口响应客户端的随机命令端口
3. ==服务端==的20号端口==主动==请求连接客户端的随机数据端口
4. 客户端的随机数据端口进行确认

- 被动模式

1. 客户端打开大于1023的随机命令端口和大于1023的随机数据端口向服务的的21号端口发起请求
2. 服务端的21号命令端口响应客户端的随机命令端口
3. ==客户端主动==连接服务端打开的大于1023的随机端口
4. 服务端进行确认
```

27.请简要说明ssh免密登陆过程

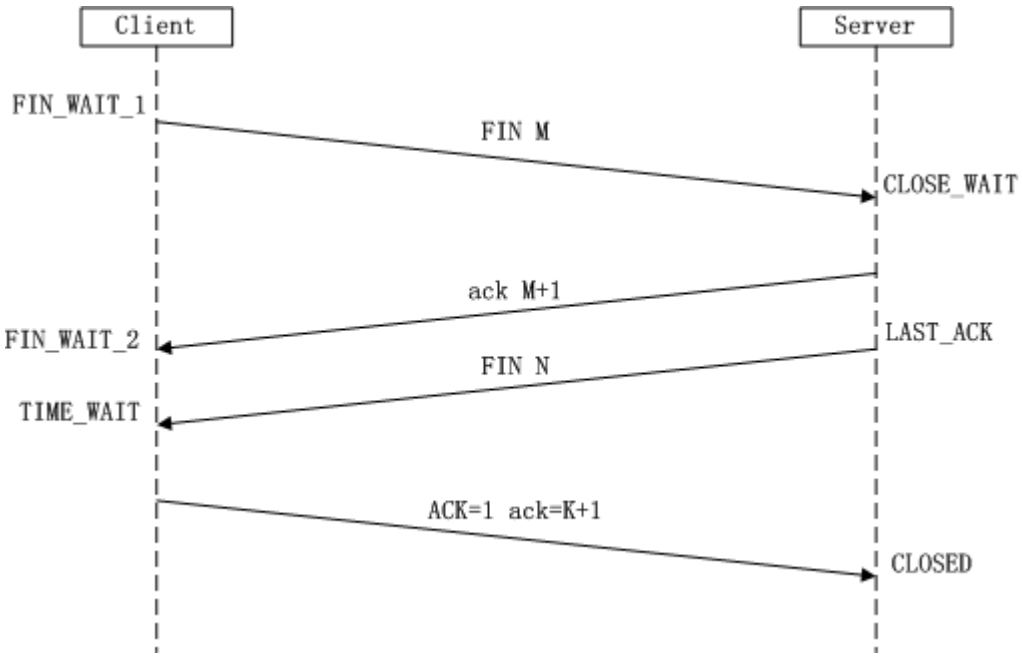


1.在server A上生成公钥私钥。
2.将公钥拷贝给server B，要重命名成authorized_keys
3.Server A向Server B发送一个连接请求。
4.Server B得到Server A的信息后，在authorized_key中进行比对，如果有相应的用户名和IP，则随机生成一个字符串，并用Server A的公钥加密，发送给Server A。
5.Server A得到Server B发来的消息后，使用私钥进行解密，然后将解密后的字符串发送给Server B。Server B进行和生成的对比，如果一致，则允许免登录。

28.tcp的3次握手和4次挥手的全进程



（1）第一次握手：Client将标志位SYN置为1，随机产生一个值seq=J，并将该数据包发送给Server，Client进入SYN_SENT状态，等待Server确认。
（2）第二次握手：Server收到数据包后由标志位SYN=1知道Client请求建立连接，Server将标志位SYN和ACK都置为1，ack=J+1，随机产生一个值seq=K，并将该数据包发送给Client以确认连接请求，Server进入SYN_RCVD状态。
（3）第三次握手：Client收到确认后，检查ack是否为J+1，ACK是否为1，如果正确则将标志位ACK置为1，ack=K+1，并将该数据包发送给Server，Server检查ack是否为K+1，ACK是否为1，如果正确则连接建立成功，Client和Server进入ESTABLISHED状态，完成三次握手，随后Client与Server之间可以开始传输数据了。



（1）第一次挥手：Client发送一个FIN，用来关闭Client到Server的数据传送，Client进入FIN_WAIT_1状态。
（2）第二次挥手：Server收到FIN后，发送一个ACK给Client，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号），Server进入CLOSE_WAIT状态。
（3）第三次挥手：Server发送一个FIN，用来关闭Server到Client的数据传送，Server进入LAST_ACK状态。
（4）第四次挥手：Client收到FIN后，Client进入TIME_WAIT状态，接着发送一个ACK给Server，确认序号为收到序号+1，Server进入CLOSED状态，完成四次挥手。

29.请写出**http**和**https**请求的区别，并写出遇到过的响应状态码

1.https协议需要到ca申请证书，一般免费证书很少，需要交费。
2.http是超文本传输协议，信息是明文传输，https 则是具有安全性的ssl加密传输协议。
3.http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。
4.http的连接很简单，是无状态的；HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全。

状态码常用：
301 永久重定向
403 服务器已经理解请求，但是拒绝执行
404 页面丢失
500 服务器错误

30.操作系统内存调度方式有哪几种并简单说明

OPT：最佳替换算法（optional replacement）。替换下次访问距当前时间最长的页。opt算法需要知道操作系统将来的事件，显然不可能实现，只作为一种衡量其他算法的标准。

LRU:最近最少使用(Least Recently Used).替换上次使用距离当前最远的页。根据局部性原理：替换最近最不可能 访问到的页。性能最接近OPT，但难以实现。可以维护一个关于访问页的栈或者给每个页添加最后访问的时间标签，但开销都很大。

FIFO:先进先出(First In First Out),将页面看做一个循环缓冲区，按循环方式替换。

Clock：时钟替换算法（Clock）,给每个页帧关联一个使用位。当该页第一次装入内存或者被重新访问到时，将使用位置为1。每次需要替换时，查找使用位被置为0的第一个帧进行替换。

31.简述**DNS**进行域名解析的过程？

用户要访问www.baidu.com，会先找本机的host文件，再找本地设置的DNS服务器，如果也没有的话，就去网络中找根服务器，根服务器反馈结果，说只能提供一级域名服务器.cn，就去找一级域名服务器，一级域名服务器说只能提供二级域名服务器.com.cn,就去找二级域名服务器，二级域服务器只能提供三级域名服务器.baidu.com.cn，就去找三级域名服务器，三级域名服务器正好有这个网站www.baidu.com，然后发给请求的服务器，保存一份之后，再发给客户端

【shell编程篇】

32.描述**Linux shell**中单引号、双引号及不加引号的简单区别

单引号：所见即所得，即将单引号内的内容原样输出，或者描述为单引号里面看到的是什么就输出什么。

双引号：把双引号里面的内容给输出出来，如果内容中有命令、变量等，会先把，变来那个、命令解析出结果，然后输出最终内容。

无引号：把内容输出出来，可能不会键含有空格的字符串，视为一个整体输出，如果内容中有命令、变量等，会先把变量、命令解析出来，然后输出最终内容，如果字符串中带有空格等特殊字符，则不能完整输出，需要改加双引号。一般连续的字符串，数字，路径等可以用，不过最好用双引号，替代之。

33.写一个脚本将某目录下大于**100k**的文件移动至**/tmp**下

```
for i in `find /test -type f -size +100k`;do cd /test && mv $i /tmp;done
```

34.写一个脚本进行**nginx**日志统计，得到访问**ip**最多的前**10**个**(nginx**日志路径：**/home/logs/nginx/default/access.log**

```
awk ‘{a[$1]++}END{for (j in a) print a[j],j}’ /home/logs/nginx/default/access.log|sort -nr|head -10
```

35.写一个脚本把指定文件里的**/usr/local**替换为别的目录

```
sed ‘s:/user/local:/tmp:g’ filename
```

36.写一个脚本，实现批量添加**20**个用户，用户名为**user01-20**，密码为**user**后面跟**5**个随机字符

```
#!/bin/bash
#description: useradd
for i in `seq -f"%02g" 1 20`;do
useradd user$i
echo "user$i-`echo $RANDOM|md5sum|cut -c 1-5`"|passwd -stdinuser$i >/dev/null 2>&1
done
```

37.写一个脚本，实现判断**192.168.1.0/24**网络里，当前在线的**IP**有哪些，能**ping**通则认为在线

```
#!/bin/bash
for ip in `seq 1 255`
do
ping -c 1 192.168.1.$ip > /dev/null 2>&1
if [ $? -eq 0 ]; then
echo 192.168.1.$ip UP
else
echo 192.168.1.$ip DOWN
fi
}&
done
wait
```

38.写一个脚本，判断一个指定的脚本是否是语法错误；如果有错误，则提醒用户键入**Q**或者**q**无视错误并退出其它任何键可以通过**vim**打开这个指定的脚本

```
#!/bin/bash
read -p "please input check script-> " file
if [ -f $file ]; then
sh -n $file > /dev/null 2>&1
if [ $? -ne 0 ]; then
read -p "You input $file syntax error,[Type q to exit or Type vim to edit]" answer
case $answer in
q | Q)
exit 0
;;
vim )
vim $file
;;
*)
exit 0
;;
esac
fi
else
echo "$file not exist"
exit 1
fi
```

39.写脚本将当前目录所有文件扩展名改为**log**

```
for file in `ls ./ | grep -P "(.*)(\..*)"`;
do
echo $file | mv $file `echo ${file%.**}`.log;
done
```

【数据库服务篇】

40.写一个脚本将数据库备份并打包至远程服务器**192.168.1.1/backup**目录下

```
mount 192.168.1.1:/backup /mnt
cd /mnt
/usr/local/mysql/bin/mysqldump -hlocalhost -uroot test >test.sql
tar czf test.sql.tar.gz test.sql
rm -f test.sql
```

41.请写**mysql**数据库中的**SQL**查询，查找**customer**表中**uid**列内大于**100**的记录并以**uid**排序，正序输出前**10**条记录

```
select * from customer where uid > 100 order by uid asc limit 10
```

42.数据库读写分离有什么好处

```
1. 将读操作和写操作分离到不同的数据库上，减轻了数据访问的压力，避免出现性能瓶颈；
2. 主服务器进行写操作时，不影响查询应用服务器的查询性能，降低阻塞，提高并发；
3. 数据拥有多个容灾副本，提高数据安全性，同时当主服务器故障时，可立即切换到其他服务器，提高系统可用性；
```

43.MongoDB与**mysql**有什么区别

```
①MongoDB在不指定_id的情况下插入速度很快，对系统内存利用率很高，适合那些读作业任务很重的任务模型，但是稳定性不如mysql。
②适合数据具体格式不明确的（弱数据结构）的情况下，对开发者友好。但是事务关系支持薄弱（nosql通病）
③自带分布式文件系统，可以很方便的部署到集群上。
④shard（数据实际存储）概念，每增加一个shard，插入性能会以接近倍数的方式增长，磁盘容量也可以方便的扩充。
⑤shard 划分 包括：mongos（接收client需求，shard集群接口）config （元数据或者配置信息，如每一个切片的片键范围，mongos会根据config信息去取数据）
```

44.redis与**memcache**区别以及应用场景

①redis支持更丰富的数据类型，共有5种（字符串，列表，hash，集合，有序集合），memcache只支持简单的字符串
②redis可以数据持久化，它不仅会将数据放在内存中，还可以存到磁盘里部分，memcache会将数据存在内存中，最大存储量即内存大小
③redis可以做主从
④redis最大可以存1G，memcache最大1M

redis应用场景
①session ：可以做持久化
②全页缓存（FPC）：持久化，重启后，用户看到的网页速度依然不会变慢
③list和set可以让redis用来做消息队列
⑤排行榜，set 和zset可以很好地进行数字排序
⑥发布/订阅功能

redis 注意事项：
①redis的主最好不要做持久化，可以对从机做（AOF）持久化，策略为每秒一次
②为了主从稳定，最好主从在同一个网段内
③尽量避免在压力过大的redis主上添加从机
④主从复制，不要用图形结构，用单链表更加稳定，这样方便解决单点故障问题，如果主挂了，直接将slave1提升为主

45.redis的数据结构

string----字符串类型，使用场景：做缓存，计数器，共享session
hash----哈希类型，使用场景：用户信息存储
list----列表类型，使用场景：消息队列，微博TimeLine
set----集合类型，使用场景：好友推荐
Sorted Set----有序集合类型，使用场景：排行榜

46.生产环境redis的开源高可用方案有哪些？

1.Twemproxy，大概概念是，它类似于一个代理方式，使用方法和普通redis无任何区别，设置好它下属的多个redis实例后，使用时在本需要连接redis的地方改为连接twemproxy，它会以一个代理的身份接收请求并使用一致性hash算法，将请求转接到具体redis，将结果再返回twemproxy。使用方式简便(相对redis只需修改连接端口)，对旧项目扩展的首选。 问题：twemproxy自身单端口实例的压力，使用一致性hash后，对redis节点数量改变时候的计算值的改变，数据无法自动移动到新的节点。
2.Codis，目前用的最多的集群方案，基本和twemproxy一致的效果，但它支持在 节点数量改变情况下，旧节点数据可恢复到新hash节点。
3.Redis cluster3.0自带的集群，特点在于他的分布式算法不是一致性hash，而是hash槽的概念，以及自身支持节点设置从节点。具体看官方文档介绍。
4.在业务代码层实现，起几个毫无关联的redis实例，在代码层，对key 进行hash计算，然后去对应的redis实例操作数据。 这种方式对hash层代码要求比较高，考虑部分包括，节点失效后的替代算法方案，数据震荡后的自动脚本恢复，实例的监控，等等。
5.Redis哨兵，Redis Sentinal着眼于高可用，在master宕机时会自动将slave提升为master，继续提供服务。

47.Redis是单线程的，如何提高多核CPU的利用率？

可以在同一个服务器部署多个Redis的实例，并把他们当作不同的服务器来使用，在某些时候，无论如何一个服务器是不够的， 所以，如果你想使用多个CPU，你可以考虑一下分片（shard）。

48.Redis常见性能问题和解决方案？

1.Master最好不要做任何持久化工作，如RDB内存快照和AOF日志文件
2.如果数据比较重要，某个Slave开启AOF备份数据，策略设置为每秒同步一次
3.为了主从复制的速度和连接的稳定性，Master和Slave最好在同一个局域网内
4. 尽量避免在压力很大的主库上增加从库
5. 主从复制不要用图状结构，用单向链表结构更为稳定，即：Master <- Slave1 <- Slave2 <- Slave3...
这样的结构方便解决单点故障问题，实现Slave对Master的替换。如果Master挂了，可以立刻启用Slave1做Master，其他不变

49.Redis提供了哪几种持久化方式？

RDB持久化方式能够在指定的时间间隔能对你的数据进行快照存储.

AOF持久化方式记录每次对服务器写的操作,当服务器重启的时候会重新执行这些命令来恢复原始的数据,AOF命令以redis协议追加保存每次写的操作到文件末尾.Redis还能对AOF文件进行后台重写,使得AOF文件的体积不至于过大.

如果你只希望你的数据在服务器运行的时候存在,你也可以不使用任何持久化方式.

你也可以同时开启两种持久化方式, 在这种情况下, 当redis重启的时候会优先载入AOF文件来恢复原始的数据,因为在通常情况下AOF文件保存的数据集要比RDB文件保存的数据集要完整.

最重要的事情是了解RDB和AOF持久化方式的不同,让我们以RDB持久化方式开始。

50.修改配置不重启**Redis**会实时生效吗？

针对运行实例，有许多配置选项可以通过 CONFIG SET 命令进行修改，而无需执行任何形式的重启。 从 Redis 2.2 开始，可以从 AOF 切换到 RDB 的快照持久性或其他方式而不需要重启 Redis。检索 ‘CONFIG GET *’ 命令获取更多信息。

但偶尔重新启动是必须的，如为升级 Redis 程序到新的版本，或者当你需要修改某些目前 CONFIG 命令还不支持的配置参数的时候。

51.Redis集群会有写操作丢失吗？为什么？

Redis并不能保证数据的强一致性，这意味这在实际中集群在特定的条件下可能会丢失写操作。

52.查看**Redis**使用情况及状态信息用什么命令？

```
info
```

53.Mongodb熟悉吗，一般部署几台？

一般mongodb部署主从、或者mongodb分片集群；建议3台或5台服务器来部署。MongoDB分片的基本思想就是将集合切分成小块。这些块分散到若干片里面，每个片只负责总数据的一部分。 对于客户端来说，无需知道数据被拆分了，也无需知道服务端哪个分片对应哪些数据。数据在分片之前需要运行一个路由进程，进程名为mongos。这个路由器知道所有数据的存放位置，知道数据和片的对应关系。对客户端来说，它仅知道连接了一个普通的mongod，在请求数据的过程中，通过路由器上的数据和片的对应关系，路由到目标数据所在的片上，如果请求有了回应，路由器将其收集起来回送给客户端。

54.Rabbitmq和**kafka**有什么区别？

1)在架构模型方面，
RabbitMQ遵循AMQP协议，RabbitMQ的broker由Exchange,Binding,queue组成，其中exchange和binding组成了消息的路由键；客户端Producer通过连接channel和server进行通信，Consumer从queue获取消息进行消费（长连接，queue有消息会推送到consumer端，consumer循环从输入流读取数据）。rabbitMQ以broker为中心；有消息的确认机制。
kafka遵从一般的MQ结构，producer，broker，consumer，以consumer为中心，消息的消费信息保存的客户端consumer上，consumer根据消费的点，从broker上批量pull数据；无消息确认机制。

2)在吞吐量，
kafka具有高的吞吐量，内部采用消息的批量处理，zero-copy机制，数据的存储和获取是本地磁盘顺序批量操作，具有O(1)的复杂度，消息处理的效率很高。
rabbitMQ在吞吐量方面稍逊于kafka，他们的出发点不一样，rabbitMQ支持对消息的可靠的传递，支持事务，不支持批量的操作；基于存储的可靠性的要求存储可以采用内存或者硬盘。

3)在可用性方面，
rabbitMQ支持miror的queue，主queue失效，miror queue接管。
kafka的broker支持主备模式。

4)在集群负载均衡方面，
kafka采用zookeeper对集群中的broker、consumer进行管理，可以注册topic到zookeeper上；通过zookeeper的协调机制，producer保存对应topic的broker信息，可以随机或者轮询发送到broker上；并且producer可以基于语义指定分片，消息发送到broker的某分片上。
rabbitMQ的负载均衡需要单独的loadbalancer进行支持。

55.备份的分类有哪些**?**

系统备份：针对整个操作系统进行备份；当操作系统损坏或者无法启动时，能通过备份快速恢复。
数据备份：针对用户的数据文件、应用软件、数据库进行备份；当这些数据丢失或损坏时，也能通过备份恢复。

56.什么是冷**/**热备份？他们各自有什么优点和缺点？

冷备份：需要备份的文档先关闭停止使用，再执行备份的方式；
优点是简单快速、容易恢复到某个时间点、方便维护；
缺点是只能恢复到某个时间点、备份期间数据不便正常使用。
热备份：指执行备份时不影响备份文档正常使用的方式；
优点是备份速度快、不影响数据使用；
缺点是所有操作都会同步，包括删除。

57.如果有一个**100G**大小的数据库该如何做备份？

100G 以上的库，可以考虑用 xtranbackup 来做，备份速度明显要比 mysqldump 要快。一般是选择一周一个全备，其余每天进行增量备份，备份时间为业务低峰期。

58.备份恢复失败如何处理**?**

首先在恢复之前就应该做足准备工作，避免恢复的时候出错。比如说备份之后的有效性检查、权限检查、空间检查等。如果万一报错，再根据报错的提示来进行相应的调整。

59.mysqldump备份的本质、优点和缺点？

本质：导出的是sql语句文件
优点：无论是什么存储引擎，都可以用mysqldump备成sql语句
缺点：速度较慢,导入时可能会出现格式不兼容的突发状况.无法直接做增量备份.

60.逻辑备份是什么？

备份的是建表、建库、插入等操作所执行SQL语句（DDL DML DCL）。
适用于中小型数据库，效率相对较低。 一般在数据库正常提供服务的前提下进行，如：mysqldump、mydumper、into outfile（表的导出导入）等。

61.物理备份是什么？

直接复制数据库文件 dbfile binary log my.cnf
适用于大型数据库环境，不受存储引擎的限制，但不能恢复到不同的MySQL版本。

62.什么是存储引擎？最常用的存储引擎有哪些？

1. 存储引擎说白了就是如何管理操作数据（存储数据、如何更新、查询数据等）的==一种方法和机制==。
2. 在MySql数据库中提供了多种存储引擎，各个存储引擎的优势各不一样。
3. 用户可以根据不同的需求为数据表选择不同的存储引擎，也可以根据自己的需要编写自己的存储引擎。
4. 甚至一个库中不同的表使用不同的存储引擎，这些都是允许的。

最常用的存储引擎是MyISAM和InnoDB。

63.什么是**mysql**的二进制日志**(binary log)**？

二进制日志记录数据库的所有更改操作（DDL/DML/DCL），不包含select或者show这类语句。
用于主从复制中，master主服务器将二进制日志中的更改操作发送给slave从服务器，从服务器执行这些更改操作是的和主服务器的更改相同。
用于数据的恢复操作。
默认二进制日志是关闭的，可以使用log-bin=xxx参数开启

64.Binlog工作模式有哪些？各什么特点，企业如何选择？

1.Row(行模式)：
日志中会记录成每一行数据被修改的形式，然后在slave端再对相同的数据进行修改

2.Statement(语句模式)
每一条修改的数据都会完整的记录到主库master的binlog里面，在slave上完整执行在master执行的sql语句

3.mixed(混合模式)
结合前面的两种模式，如果在工作中有使用函数 或者触发器等特殊功能需求的时候，使用混合模式

数据量达到比较高时候，它就会选择 statement模式，而不会选择Row Level行模式

65.如何在线正确清理**MySQL binlog**？

```
MySQL中的binlog日志记录了数据中的数据变动，便于对数据的基于时间点和基于位置的恢复
但日志文件的大小会越来越大，点用大量的磁盘空间，因此需要定时清理一部分日志信息
手工删除：
首先查看主从库正在使用的binlog文件名称
show master(slave) status\G

删除之前一定要备份，删除指定时间前的日志：
purge master logs before '2017-09-01 00:00:00';

删除指定的日志文件：
purge master logs to 'mysql-bin.000001';

自动删除：
通过设置binlog的过期时间让系统自动删除日志，查看过期时间与设置过期时间
show variables like 'expire_logs_days';
set global expire_logs_days = 30;
```

66.什么是mysql的中继日志？

```
用于主从复制，master主服务器将自己的二进制日志发送给slave从服务器，slave先保存在自己的中继日志中，然后再执行自己本地的relay log里的sql达到数据库更改和master保持一致。
默认中继日志没有开启，可以使用`relay-log`参数开启
```

67.数据库和数据库实例之间的关系是什么？

```
通常情况下，数据库实例和数据库是一一对应的关系，也就是一个数据库实例对应一个数据库； 但是，在集群环境中存在多个数据库实例共同使用一个数据库。比如：oracle RAC
```

68.什么叫mysql事务？

```
MySQL 事务主要用于处理操作量大，复杂度高的数据。比如说，在人员管理系统中，你删除一个人员，你即需要删除人员的基本资料，也要删除和该人员相关的信息，如信箱，文章等等，这样，这些数据库操作语句就构成一个事务！一般来说，事务是必须满足4个条（ACID）：原子性（Atomicity，或称不可分割性）、一致性（Consistency）、隔离性（Isolation，又称独立性）、持久性（Durability）。
```

69.MySQL密码丢了，请找回？

```
mysqld_safe --skip-grant-tables &   #启动数据库服务
mysql -uroot -ppassowrd -e "use mysql;update user set passowrd = PASSWORD('newpassword') where user = 'root';flush privileges;"
```

70.请说出非关系型数据库的典型产品、特点及应用场景？

```
1.memecached 纯内存
2.redis 持久化缓存
3.mongodb 面向文档
如果需要短时间相应的查询操作，没有良好模式定义的数据存储，或者模式更改频繁的数据存储还是用NoSQL
```

71.什么是MySQL多实例，如何配置MySQL多实例？

```
mysql多实例就是在同一台服务器上启用多个mysql服务，它们监听不同的端口，运行多个服务进程，它们相互独立，互不影响的对外提供服务，便于节约服务器资源与后期架构扩展

多实例的配置方法有两种：
1、一个实例一个配置文件，不同端口
2、同一配置文件(my.cnf)下配置不同实例，基于mysqld_multi工具
```

72.如何加强**MySQL**安全，请给出可行的具体措施？

```
1、删除数据库不使用的默认用户
2、配置相应的权限（包括远程连接）
3、不可在命令行界面下输入数据库的密码
4、定期修改密码与加强密码的复杂度
```

73.MySQL Sleep线程过多如何解决？

```
1、可以杀掉sleep进程，kill PID
2、修改配置，重启服务
[mysqld]
wait_timeout = 600
interactive_timeout=30
注意：如果生产服务器不可随便重启也可以使用下面的方法解决
set global wait_timeout=600
set global interactive_timeout=30;
```

74.sort_buffer_size参数作用？如何在线修改生效？

```
在每个connection(session)第一次连接时需要使用到，来提访问性能
 set global sort_buffer_size = 2M
```

75.MySQL中**myisam**与**innodb**的区别**?**

```
1.InnoDB支持事物，而MyISAM不支持事物

2.InnoDB支持行级锁，而MyISAM支持表级锁

3.InnoDB支持MVCC，而MyISAM不支持

4.InnoDB支持外键，而MyISAM不支持

5.InnoDB不支持全文索引，而MyISAM支持。
```

76.如何调整生产线中**MySQL**数据库的字符集？

```
1、首先导出库的表结构 -d 只导出表结构，然后批量替换
2、导出库中的所有数据（在不产生新数据的前提下）
3、然后全局替换set names = xxxxx
4、删除原有库与表，并新创建出来，再导入建库与建表语句与所有数据
```

77.请描述**MySQL**里中文数据乱码原理，如何防止乱码？

```
服务器系统、数据库、客户端三方字符集不一致导致，需要统一字符
```

78.企业生产**MySQL**如何优化（请多角度描述）？

```
1、提升服务器硬件资源与网络带宽
2、优化mysql服务配置文件
3、开启慢查询日志然后分析问题所在
```

79.生产中误操作执行了一个**drop**库**SQL**语句，如何完整恢复？

```
1、停止主从复制，在主库上执行锁表并刷新binlog操作，接着恢复之前的全备文件（比如0点的全备）

2、将0点时的binlog文件与全备到故障期间的binlog文件合并导出成sql语句
mysqlbinlog -no-defaults mysql-bin.000011 mysql-bin.000012 >bin.sql

3、将导出的sql语句中drop语句删除，恢复到数据库中
mysql -uroot -pmysql123 < bin.sql
```

80.工作中遇到过哪些数据库故障，请描述**2**个例子？

```
1、开发使用root用户在从库上写入数据造成主从数据不一致，并且前端没有展示需要修改的内容（仍旧是老数据）

2、内网测试环境服务器突然断电造成主从同步故障
```

【服务器架构篇】

81.常见的**MySQL**架构**?**

```
主从机构    主主架构      主主主架构
```

82.mysql复制基本原理流程

```
1. 主：binlog线程—记录下所有改变了数据库数据的语句，放进master上的binlog中；
2. 从：io线程—在使用start slave 之后，负责从master上拉取 binlog 内容，放进自己的relay log中；
3. 从：sql执行线程—执行relay log中的语句；
```

83.MySQL复制的线程有几个及之间的关联

```
MySQL 的复制是基于如下 3 个线程的交互：
1. Master 上面的 binlog dump 线程，该线程负责将 master 的 binlog event 传到slave；
2. Slave 上面的 IO 线程，该线程负责接收 Master 传过来的 binlog，并写入 relay log；
3. Slave 上面的 SQL 线程，该线程负责读取 relay log 并执行；
```

84.如何检测出**mysql**主从有哪些数据不一致？如何修复？

```
1、可以使用Percana-Toolkit工具组件的pt-table-checksum校验出来
2、然后在用pt-check-sync进行同步更新（根据实际情况使用）
```

85.线上**MySQL**主从复制故障如何解决？

```
登陆从库
1、执行stop slave;停止主从同步
2、然后set global sql_slave_skip_counter = 1;跳过一步错误
3、最后执行 start slave;并查看主从同步状态

需要重新进行主从同步操作步骤如下
进入主库
1、进行全备数据库并刷新binlog,查看主库此的状态
2、恢复全备文件到从库，然后执行change master
3、开启主从同步start slave;并查看主从同步状态
```

86.如何监控主从复制是否故障**?**（或者**zabbix**监控**mysql**主从复制状态是否正常的脚本思路是？）

```
mysql -uroot -ppassowrd -e "show slave status\G" |grep -E
"Slave_IO_Running|Slave_SQL_Running"|awk '{print $2}'|grep -c Yes
通过判断Yes的个数来监控主从复制状态，正常情况等于2
```

87.生产一主多从从库宕机，如何手工恢复？

```
1、执行stop slave 或者停止服务
2、修复好从库数据库
3、然后重新操作主库同步
```

88.生产一主多从主库宕机，如何手工恢复？

```
1、登陆各个从库停止同步，并查看谁的数据最新，将它设置为新主库让其它从库同步其数据
2、修复好主库之后，生新操作主从同步的步骤就可以了

#需要注意的新的主库如果之前是只读，需要关闭此功能让其可写
#需要在新从库创建与之前主库相同的同步的用户与权限
#其它从库执行change master to master_port=新主库的端口，start slave
```

89.MySQL出现复制延迟有哪些原因？如何解决？

```
1、需要同步的从库数据太多
2、从库的硬件资源较差，需要提升
3、网络问题，需要提升网络带宽
4、主库的数据写入量较大，需要优配置和硬件资源
5、sql语句执行过长导致，需要优化
6.5.6之前是单线程复制，之后是多线程，也可采用5.7以上版本
```

90.给出企业生产大型**MySQL**集群架构可行备份方案？

```
1、双主多从，主从同步的架构，然后实行某个从库专业做为备份服务器
2、编写脚本实行分库分表进行备份，并加入定时任务
3、最终将备份服务推送至内网专业服务器，数据库服务器本地保留一周
4、备份服务器根据实际情况来保留备份数据（一般30天）
```

91.讲述一下**Tomcat8005**、**8009**、**8080**三个端口的含义？

```
8005-------关闭时使用
8009-------为AJP端口，即容器使用，如Apache能通过AJP协议访问Tomcat的8009端口
8080-------一般应用使用
```

92.Tomcat有三种工作模式：Bio、Nio和Apr，他们工作原理是？

```
Bio(Blocking I/O)：默认工作模式，阻塞式I/O操作，没有任何优化技术处理，性能比较低。
Nio(New I/O or Non-Blocking)：非阻塞式I/O操作，有Bio有更好的并发处理性能。
Apr(Apache Portable Runtime，Apache可移植运行库)：首选工作模式，主要为上层的应用程序提供一个可以跨越多操作系统平台使用的底层支持接口库。
tomcat利用基于Apr库tomcat-native来实现操作系统级别控制，提供一种优化技术和非阻塞式I/O操作，大大提高并发处理能力。但是需要安装apr和tomcat-native库。
```

93.请解释 Tomcat 中使用的连接器是什么？

```
在 Tomcat 中，使用了两种类型的连接器：
HTTP 连接器 ：它有许多可以更改的属性，以确定它的工作方式和访问功能，如重定向和代理转发
AJP 连接器 ：它以与 HTTP 连接器相同的方式工作，但是他们使用的是 HTTP 的 AJP 协议。AJP 连接器通常通过插件技术 mod_jk 在 Tomcat 中实现
```

94.Tomcat调优大概有哪些思路？

```
1、增加最大连接数
2、调整工作模式
3、启用gzip压缩
4、调整JVM内存大小
5、与Apache或Nginx整合，实现动静分离
6、合理选择垃圾回收算法
7、尽量使用较新JDK版本
```

95.请写下命令检查nginx的当前配置文件，然后平滑重启

```
1.确认nginx配置文件的语法是否正确
  nginx -t
2.平滑重启
  kill -HUP nginx进程号
  kill -HUP /路径/nginx.pid
```

96.请简单描述nginx与php-fpm的两种连接方式及其优缺点

```
在linux中，nginx服务器和php-fpm可以通过tcp socket和unix socket两种方式实现。
unix socket是一种终端，可以使同一台操作系统上的两个或多个进程进行数据通信。这种方式需要再nginx配置文件中填写php-fpm的pid文件位置，效率要比tcp socket高。
tcp socket这种通信方式，需要在nginx配置文件中填写php-fpm运行的ip地址和端口号。这种方式的优点是可以跨服务器，当nginx和php-fpm不在同一台机器上时，只能使用这种方式。
```

97.你常用的Nginx模块，用来做什么

```
rewrite模块，实现重写功能
access模块：来源控制
ssl模块：安全加密
ngx_http_gzip_module：网络传输压缩模块
ngx_http_proxy_module 模块实现代理
ngx_http_upstream_module模块实现定义后端服务器列表
ngx_cache_purge实现缓存清除功能
```

98.指出**Nginx**支持哪几种负载均衡模式，并指出各模式的应用场景

```
1.roundrobin 轮询方式，依次将请求分配到各个后台服务器中，默认的负载均衡方式。 适用于后台机器性能一致的情况。 挂掉的机器可以自动从服务列表中剔除。
2.weight 根据权重来分发请求到不同的机器中，适用于后台机器性能不一样的情况。
3.ip_hash 根据请求者ip的hash值将请求发送到后台服务器中，可以保证来自同一ip的请求被打到固定的机器上，可以解决session问题。
4.url_hash 根据请求的url的hash值将请求分到不同的机器中，当后台服务器为缓存的时候效率高。
5.fair 根据后台响应时间来分发请求，响应时间短的分发的请求多。
```

99.讲一下**Keepalived**的工作原理？

```
在一个虚拟路由器中，只有作为MASTER的VRRP路由器会一直发送VRRP通告信息,BACKUP不会抢占MASTER，除非它的优先级更高。当MASTER不可用时(BACKUP收不到通告信息),多台BACKUP中优先级最高的这台会被抢占为MASTER。这种抢占是非常快速的(<1s)，以保证服务的连续性,由于安全性考虑，VRRP包使用了加密协议进行加密。BACKUP不会发送通告信息，只会接收通告信息
```

100.keepalived的主要模块有哪些？

```
keepalived主要有三个模块，分别是core、check和vrrp。
core模块为keepalived的核心，负责主进程的启动、维护及全局配置文件的加载和解析。
check负责健康检查，包括常见的各种检查方式
vrrp模块是来实现VRRP协议的
```

101.keepalived如何做到健康检查**?**

```
Keepalived健康检查方式配置
HTTP_GET|SSL_GET
HTTP_GET | SSL_GET
{
url {
path /                  # HTTP/SSL 检查的url可以是多个
digest <STRING>         # HTTP/SSL 检查后的摘要信息用工具genhash生成
status_code 200         # HTTP/SSL 检查返回的状态码
}
connect_port 80         # 连接端口
bindto<IPADD>
connect_timeout 3       # 连接超时时间
nb_get_retry 3          # 重连次数
delay_before_retry 2    #连接间隔时间
}
```

102.Keepalived怎么实现高可用？

Keepalived高可用服务对之间的故障切换转移，是通过VRRP协议来实现的。
在 Keepalived服务正常工作时，主 Master节点会不断地向备节点发送（多播的方式）心跳消息，用以告诉备Backup节点自己还活看，当主 Master节点发生故障时，就无法发送心跳消息，备节点也就因此无法继续检测到来自主 Master节点的心跳了，于是调用自身的接管程序，接管主Master节点的 IP资源及服务。而当主 Master节点恢复时，备Backup节点又会释放主节点故障时自身接管的IP资源及服务，恢复到原来的备用角色。

103.Linux集群主要有哪几类？

1.LB 负载均衡集群
负载均衡集群主要是提高服务的响应能力，如果一组计算机节点（或者一组进程）提供相同的（同质的）服务，那么对服务的请求就应该均匀的分摊到这些节点上。软件级的比较常见的有如下两种:LVS、Haproxy
2.HA 高可用性集群
高可用性集群主要是提供7*24小时不间断服务的,如果某台宕机了,会自动的切换到其他计算机上面工作,从而达到高可用的效果。HA 高可用集群的解决方案常见的有以下几种heartbeat、corosync+openais
RHCS、ultramokey、keepalived
3.HP 高性能集群
高性能集群主要是用于需要大量CPU运算的场景中,比如说天气预报,国外3D大片的特效制作,等等一系列需要做大量运算的应用, HP 高性能集群的解决方案常见的有bowerful

104.常见几种负载均衡方式的比较？

一、LVS的特点
1、工作在网络4层上，抗负载能力强，作分发之用；
2、配置性比较低；
3、工作稳定，自身具备的双机热备方案；
4、应用范围比较广，可以对所有应用做负载均衡；

二、NGINX的特点
1、工作在网络的7层之上；
2、对网络的依赖比较小；
3、安装和配置比较简单，测试起来比较方便；
4、可以承担高的负载压力且稳定；
5、可以通过端口检测到服务器内部的故障，
6、对请求的异步处理可以帮助节点服务器减轻负载；
7、能支持http和Email；
8、默认的只有Round-robin和IP-hash两种负载均衡算法；

三、Haproxy的特点
1、工作在网络7层之上。
2、能够补充Nginx的一些缺点比如Session的保持，Cookie的引导等工作
3、支持url检测后端的服务器出问题的检测
4、更多的负载均衡策略
5、有更出色的负载均衡速度
6、HAProxy可以对Mysql进行负载均衡，对后端的DB节点进行检测和负载均衡

105.lvs调度算法中的最小连接数算法原理是什么

最少连接调度算法是把新的连接请求分配到当前连接数最小的服务器，最小连接调度是一种动态调度短算法，它通过服务

器当前所活跃的连接数来估计服务器 的负载均衡，调度器需要记录各个服务器已建立连接的数目，当一个请求被调度到某台服务器，其连接数加1，当连接中止或超时，其连接数减一，在系统实现时， 我们也引入当服务器的权值为0时，表示该服务器不可用而不被调度。

106.简单介绍**lvs**的三种工作模型

1、NAT模型
NAT模型是通过网络地址转换来实现的,工作方式是,首先用户请求到达前端的负载均衡器，然后负载均衡器根据事先定义好的调度算法将用户请求的目标地址修改为后端的应用服务器，应用程序服务器处理好请求之后将结果返回给用户,期间必须要经过负载均衡器,负载均衡器将报文的源地址 改为用户请求的目标地址,再转发给用户,从而完成整个负载均衡的过程,
2、DR模型
DR模型是通过路由技术实现的负载均衡技术,这种模型与NAT模型不同的地方是,负载均衡器通过改写用户请求报文中的MAC地址,将请求发送到 Real Server, 而Real Server直接响应用户,这样就大大的减少负载均衡器的压力,DR模型也是用的最多的一种。
3、TUN模型
TUN模型是通过IP隧道技术实现的,TUN模型跟DR模型有点类似,不同的地方是负载均衡器(Director Server)跟应用服务器(Real Server)通信的机制是通过IP隧道技术将用户的请求转发到某个Real Server,而Real Server 也是直接响应用户的

107.集群中资源隔离的解决方案？

1、当集群分裂成两个小集群时会发生资源争用的情况，为避免争用后端存储系统而造成灾难性的系统崩溃，集群系统引入了投票机制，只有拥有半数以上合法票数的集群才能存活，否则就推出集群系统。
2、当集群为偶数时，如果分裂，两边可能都掌握相等的票数；因此，集群系统不应该为偶数，如果是偶数则需要一个额外的ping节点参与投票。
3、票数不足的集群退出集群服务后，为了保证它不会争用资源需要STONITH机制来进行资源隔离。

【存储篇】

108.简单介绍**raid**的级别？

Raid级别	描述
raid0	读写性能佳，坏了其中一块，数据挂掉，可靠性低，磁盘利用率100%
raid1	镜像备份,同一份数据完整的保存在多个磁盘上，写的性能不佳，可靠性高，读的性能还行，磁盘利用率50%
raid5	由多块磁盘做raid 5，磁盘利用率为n-1/n, 其中一块放校验数据，允许坏一块盘，数据可以利用校验值来恢复
raid6	在raid5的基础上再加一块校验盘，进一步提高数据可靠性
raid10	先做raid 1 再做raid 0

109.简述**raid0 raid1 raid5** 三种工作模式的工作原理及特点

RAID 0：带区卷，连续以位或字节为单位分割数据，并行读/写于多个磁盘上，因此具有很高的数据传输率
但它没有数据冗余，RAID 0只是单纯地提高性能，并没有为数据的可靠性提供保证,而且其中的一个磁盘失效将影响到所有数据 因此，RAID 0 不能应用于数据安全性要求高的场合

RAID 1：镜像卷，它是通过磁盘数据镜像实现数据冗余，在成对的独立磁盘上产生互为备份的数据
不能提升写数据效率。当原始数据繁忙时，可直接从镜像拷贝中读取数据，因此RAID1 可以提高读取性能
RAID 1 是磁盘阵列中单位成本最高的，镜像卷可用容量为总容量的1/2，但提供了很高的数据安全性和可用性。 当一个磁盘失效 时，系统可以自动切换到镜像磁盘上读写，而不需要重组失效的数据

RAID 5：至少由3块硬盘组成，分布式奇偶校验的独立磁盘结构，它的奇偶校验码存在于所有磁盘上
任何一个硬盘损坏，都可以根据其它硬盘上的校验位来重建损坏的数据（最多允许1块硬盘损坏）,所以raid5可以实现数据冗余， 确保数据的安全性，同时raid5也可以提升数据的读写性能

110.网站有存储下载文件的需求，用什么存储文件比较好

```
glasterfs    nfs
```

111.简单介绍**glasterfs**的特点

glasterfs是无元数据服务器设计，没有单点故障和性能瓶颈，有很好的扩展性，和稳定性，认为存储是软件的事，不能局限于硬件。以原始数据的形式存储，访问数据简单，迁移容易。有的视频公司将他作为片库。

缺点：数据一致性问题复杂，文件目录遍历效率低，缺乏全局监控，客户端负载大，占用了大量cpu和内存，用户空间效率低，与内核空间要经常交换数据，借用FUSE，有性能损耗

112.还有什么比较好的文件存储方式

```
ceph
```

113.DAS、NAS、SAN使用场景

DAS：存储设备通过scsi或fc接口直连到服务器，数据与操作系统未分离。
NAS：将存储功能与服务器分离开，用以太网存取数据，有自己的文件管理系统，是文件级存储。多适用于文件服务器，用来存储非结构化数据。
SAN：是通过光纤交换机连接存储阵列和服务器，建立专用数据存储的存储私网。每台服务器独自管理文件。SAN允许任何服务器连接到任何存储阵列（好处是：不管数据放在哪里，服务器都可以直接存取所需的数据哦）适用于大型应用或数据库系统，缺点是成本高、较为复杂。
通俗一点，在linux里lsblk，看到的就是SAN提供的块设备，windows上的共享文件夹，一般是NAS的NFS服务。

114.存储类型的分类有哪几种？并简单进行描述

存储类型分类	描述
文件存储	NAS都属于这一类。简单来说就是mount后直接使用的。
块存储	SAN都属于这一类。简单来说就是类似/dev/sdb这种，要分区,格式化后才能mount使用
对象存储	简单来说，就是直接使用程序接口去访问

115.请你对各个存储类型的优缺点进行比较？

文件存储:类似一个大的目录，多个客户端都可以挂载过来使用
优点：利于数据共享
缺点：速度较慢
块存储：类似一个block设备，客户端可以格式化，挂载并使用，和用一个硬盘一样。
优点：和本地硬盘一样,直接使用
缺点：数据不共享
对象存储:一个对象我们可以看成一个文件,综合了文件存储和块存储的优点
优点：速度快,数据共享
缺点：成本高，不兼容现有的模式

116.什么是分布式存储?

分布式存储可以看作拥有多台存储服务器连接起来的存储导出端（多对一，多对多)。把这多台存储服务器 的存储合起来做成一个整体再通过网络进行远程共享,共享的方式有目录(文件存储),块设备(块存储),对象网关或者说 一个程序接口(对象存储)。
常见的分布式存储开源软件有:GlusterFS,Ceph,HDFS,MooseFS,FastDFS等

117.分布式存储的优点有哪些?

分布式存储一般都有以下几个优点:
1. 扩容方便，轻松达到PB级别或以上
2. 提升读写性能或数据高可用
3. 避免单个节点故障导致整个架构问题
4. 价格相对便宜，大量的廉价设备就可以组成，比光纤SAN这种便宜很多。

【Python编程篇】

118.写脚本生成随机的**20**个**ID**

ID格式要求：时间戳三位随机数字号码8位随机小写字母1506571959089xxkeabef

```
#!/usr/bin/python
import datetime
idlist =[]
for _ in range(20):
    s1=datetime.datetime.now().timestamp()    #返回的是时间戳，但是带微秒
    s2=".join([str(random.randint(0,9)) for _ in range(3)])"
    s3=".join([chr(random.randint(97,122)) for _ in range(8)])"
    idlist.append(str(int(s1))+'_'+s2+'_'+s3)
print(idlist)
```

119.写脚本判断密码强弱

```
要求密码必须由 10-15位 指定字符组成：
十进制数字，大写字母，小写字母，下划线，要求四种类型的字符都要出现才算合法的强密码
例如：Aatb32_67mnq，其中包含大写字母、小写字母、数字和下划线，是合格的强密码

#!/usr/bin/python
s=input("请输入密码： ")
count=0
flag1,flag2,flag3,flag4=True,True,True,True
len=len(s)
if len>= 10 and len<=15:
    for i in s:
        if i in "0123456789":
            if flag1:
                count+=1
            flag1=False
        if i in "ABCDEFGHIJKLMNOPQRSTUVWXYZ":
            if flag2:
                count+=1
            flag2=False
        if i in "abcdefghijklmnopqrstuvwxyz":
            if flag3:
                count+=1
            flag3=False
        if i in "_":
            if flag4:
                count+=1
            flag4=False
    if count==4:
        print("it's a right passwd")
    else:
        print("passwd is wrong")
else:
        print("the length is wrong")
```

120.写脚本列举当前目录以及所有子目录下的文件，并打印出绝对路径

```
#!/usr/bin/env python
import os
for root,dirs,files in os.walk('/tmp'):
    for name in files:
        print (os.path.join(root,name))
os.walk()
```

121.写脚本生成磁盘使用情况的日志文件

```
#!/usr/bin/env python
import time
import os
new_time = time.strftime('%Y-%m-%d')
disk_status = os.popen('df -h').readlines()
str1 = ''.join(disk_status)
f = file(new_time+'.log','w')
f.write('%s' % str1)
f.flush()
f.close()
```

122.写脚本统计出每个**IP**的访问量有多少？（从日志文件中查找）

```
#!/usr/bin/python
list = []
f = file('/usr/local/nginx/logs/access.log')
str1 = f.readlines()
f.close()
for i in str1:
        ip = i.split()[0]
        list.append(ip)
        list_num = set(list)
for j in list_num:
        num = list.count(j)
        print '%s : %s' %(j,num)
```

【运维自动化篇】

123.写出最少**3**个监控系统指标的命令（如内存，**CPU** ，**IO**，磁盘等）

```
看内存：free
看cpu：more /proc/cpuinfo
看IO：iostat -x 10
看磁盘：fdisk -l
```

124.zabbix 自定义发现是怎么做的

```
1、首先需要在模板当中创建一个自动发现的规则，这个地方只需要一个名称和一个键值。
2、过滤器中间要添加你需要的用到的值宏。
3、然后要创建一个监控项原型，也是一个名称和一个键值。
4、然后需要去写一个这样的键值的收集。

自动发现实际上就是需要首先去获得需要监控的值，然后将这个值作为一个新的参数传递到另外一个收集数据的item里面去。
```

125.zabbix 是怎么微信报警的

```
1、首先，需要有一个微信企业号。（一个实名认证的[微信号]一个可以使用的[手机号]一个可以登录的[邮箱号]
2、下载并配置微信公众平台私有接口。
3、配置Zabbix告警，（增加示警媒介类型，添加用户报警媒介，添加报警动作）
```

126.zabbix 怎么开启自定义监控

1、写一个脚本用于获取待监控服务的一些状态信息。
2、在zabbix客户端的配置文件zabbix_agentd.conf中添加上自定义的“UserParameter”，目的是方便zabbix调用我们上面写的那个脚本去获取待监控服务的信息。
3、在zabbix服务端使用zabbix_get测试是否能够通过第二步定义的参数去获取zabbix客户端收集的数据。
4、在zabbix服务端的web界面中新建模板，同时第一步的脚本能够获取什么信息就添加上什么监控项，“键值”设置成前面配置 的“UserParameter”的值。
5、数据显示图表，直接新建图形并选择上一步的监控项来生成动态图表即可。

127.zabbix 监控了多少客户端 客户端是怎么进行批量安装的

批量安装步骤：
1、使用命令生成密钥。
2、将公钥发送到所有安装zabbix客户端的主机。
3、安装 ansible 软件，（修改配置文件，将zabbix 客户机添加进组）。
4、创建一个安装zabbix客户端的剧本。
5、执行该剧本。
6、验证。

128.jenkins你都用了哪些插件？

ssh remote hosts 这个可以在远程服务器上面执行脚本。
Role Strategy Plugin 用来精细化管理权限。
SCM: 除CVS和Subversion外需要实现与源代码控制系统支持的插件。
Triggers: 事件监听并触发构建的插件。例如，URL改变触发器将监控一个URL；当地址内容发生改变，这个触发器就将执行一次作业。
Build tools: 实现额外构建工具的插件，如MSBuild和Rake。如果您想在Hudson中构建非Java的软件时这些就特别有用。
Build wrappers: 通常涉及时执行在受控制的构建过程本身之前和之后事件的插件。例如， VMware插件将在构建之前启动一个客户虚拟机，建立和然后在构建完成后关闭它。这在您可能需要访问VM以执行单元测试的情况下是非常有用的。

129.jenkins如何进行代码发布和回滚？

发布：jenkins配置好代码路径（SVN或Gitlab），然后拉代码，打tag。需要编译就编译，编译之后推送到发布服务器（jenkins里面可以调脚本），然后从分发服务器往下分发到业务服务器上。

回滚：按照版本号到发布服务器找到对应的版本推送

130.自动化运维工具**ansible**的特性有哪些？列举一些常用的模块？

no agent ，不需要安装客户端（支持ssh），no server 不需要启动服务（ansible），基于模块工作，可以使用任意语言开发模块，基于ssh工作(基于密钥认证)，YAML格式，编排任务，支持丰富的数据结构（剧本playbook），使用python编写，维护简单
常用的模块有：ping、user、group、file、shell、script、copy、yum、service等

131.Ansbile工具的**shell**、**script**模块的区别？

script模块 [在远程主机执行主控端的shell/python脚本]
shell模块 [执行远程主机上的shell/python脚本]

【云计算篇】

132.写出**hadoop**集群常用进程以及进程含义

1.写出hadoop集群常用进程以及进程含义
1、Namenode
它是Hadoop 中的主服务器，管理文件系统名称空间和对集群中存储的文件的访问。
2、Datanode
它负责管理连接到节点的存储（一个集群中可以有多个节点）。每个存储数据的节点运行一个 datanode 守护进程。
3、secondaryNameNode
它不是 namenode 的冗余守护进程，而是提供周期检查点和清理任务。 出于对可扩展性和容错性等考虑，我们一般将SecondaryNameNode运行在一台非NameNode的机器上。
4、ResourceManager
负责调度 DataNode上的工作。每个 DataNode有一个NodeManager，它们执行实际工作。
5、NodeManager
负责执行ResourceManager分发的任务

133.什么是云计算？

它是以虚拟化技术为核心技术和基础，面向服务架构(SOA)的一种实现,将虚拟化环境"资源池"隐藏起来，将其上层应用软件形成丰富的云管理接口，达到所有人自由使用所有资源的一种现象，他是一种资源使用模式的变革。

134.云计算的实现方式有哪些？

云计算的实现方式有：Private、IaaS、PaaS、SaaS
Private：传统/私有方式
　　优点：所有事情都亲自做，可控
　　缺点：用户成本比较高，要求自身技术水平高
　　典型软件：传统物理机
IaaS：基础设施即服务
　　优点：底层硬件到操作系统，都不需要用户操心，可以集中精力做业务项目
　　缺点：服务商提供的东西，不能自己自由定制，不可控
　　典型软件：OpenStack、CloudStack
PaaS：平台即服务
　　优点：对于只会开发不会运维的人员比较友好，底层到运行环境，都不需要用户操心，可以集中精力做应用项目
　　缺点：服务商提供的东西，不灵活，只适用于特殊的应用项目
SaaS：软件、应用即服务
　　优点：所有东西都由服务商提供，用户只需要花钱就行，对于广大企业来说，SaaS是采用先进技术实施信息化的最好途径。比如说，买　　企业邮箱，买财务软件云
　　缺点：对客户来说，所有的东西都不可控，安全性不够。

135.如果批量创建多个**VM**实例，是同时创建还是按顺序创建呢？

为了避免同时创建多个VM实例时候，给用户和各种资源带来的压力，我们应该按照创建请求的顺序，一个一个的创建，而满足顺序的功能的软件，这就可以用异步协作的消息队列。

136.什么是虚拟化？

虚拟化是一种技术，它的目的在于提高资源的使用率，并将底层硬件和上层的应用软件进行隔离，使得上层软件及应用计算变得更加弹性可控。最终达到有限成本的高价值。
默认情况下，虚拟化技术默认并不对外体用抽象的上层应用软件服务组件，一个没有被服务化的虚拟化环境只能被称为"资源池"，只有内部管理人员才可以操作。

137.虚拟化和云计算的区别？

虚拟化是一种技术，云计算是资源交付模式，云计算不等于虚拟化。
云计算是基于虚拟化技术的一种资源交付使用模式。

138.OpenStack的组件有哪些？

```
Cinder：为VMs提供持久的块存储能力，支持多种存储方式，工作中ceph用的比较多
Glance：用于存储和检索磁盘映像文件，支持多种存储方式
Heat：openstack 的 任 务 编 排 工 具
Horizon：openstack 的 web 可 视 化 界 面
Keystone：为Openstack中的所有服务提供了认证、授权以及端点编录服务员
Nova：管 理 VM 的 所 有 操 作
Netron：为Openstack提供网络的功能；插件化设计，支持众多流行的网络
Swift：分布式存储，基于RESTful的API实现非结构化数据对象的存储及检索
Trove：提供数据库即服务的功能
sahara：在OpenStack中提供大数据服务，生产可用
Octavia：openstack中的负载均衡项目，生产可用。
IRonic，物理裸机管理，目前是非常好用。
Ceilometer，用于实现监控和计量服务的实现，缺乏后续发展
```

139.OpenStack的核心服务有哪些？

```
compute、networking、storage、dashboard
```

140.容器退出后，通过**docker ps** 命令查看不到，数据会丢失么？

容器退出后会处于终止（exited）状态，此时可以通过 docker ps -a 查看，其中数据不会丢失，还可以通过docker start 来启动，只有删除容器才会清除数据。

141.如何控制容器占用系统资源（**CPU**，内存）的份额？

在使用docker create命令创建容器或使用docker run 创建并运行容器的时候，可以使用-c|-cpu-shares[=0]参数来调整同期使用CPU的权重，使用-m|-memory参数来调整容器使用内存的大小。

142.如何更改**Docker**的默认存储设置？

Docker的默认存放位置是/var/lib/docker,如果希望将Docker的本地文件存储到其他分区，可以使用Linux软连接的方式来做。

143.Docker公司的三款用于解决多容器分布式软件可移植部署的问题，推出的编排工具有哪些？

1.Docker Machine：为本地私有数据中心及公有云平台提供Docker引擎，实现从零到Docker的一键部署。
2.Docker Compose：是一个编排多容器分布式部署的工具，提供命令集管理容器化应用的完整开发周期，包括服务构建，启动和停止。
3.Docker Swarm：为Docker容器提供了原生的集群，它将多个Docker引擎的资源汇聚在一起，并提供Docker标准的API，使Docker可以轻松扩展到多台主机。

144.简单描述Docker-compose编排和管理多容器的过程？

1.使用Dockerfile定义应用依赖的镜像
2.使用docker-compose.yml定义应用具有的服务
3.通过docker-compose up命令创建并运行应用