注意事项？

不好理解的概念让chat通俗易懂地解释

举例子，比喻更容易理解概念

可以作为学习笔记，通过问题形式记录

无论大小，都可以记录

可以先把问题记录，写在其它笔记软件上

学任何东西都可以在这里对自己提问

要单独做笔记的可以单独记录

充当随记的功能，记录一些有价值的信息，里面包含了自己的一些思考等

都要以提问题的形式：是什么、怎么样、为什么、凭什么…

以问号结尾

锻炼自己问问题的能力，能够提出问题，有自己的想法，如果要单独做笔记可以单独去做，不影响这个

学习一定要输出，自己能提出问题，然后回答，输出才会有印象，也可以去网上找面经和面试题拓宽思路。

什么是消息队列？

消息： 跨进程传递的数据

队列：一种先进先出的数据结构

什么是中间件？

中间件是软件

提供系统软件和应用软件之间的连接

提供软件各部件之间连接

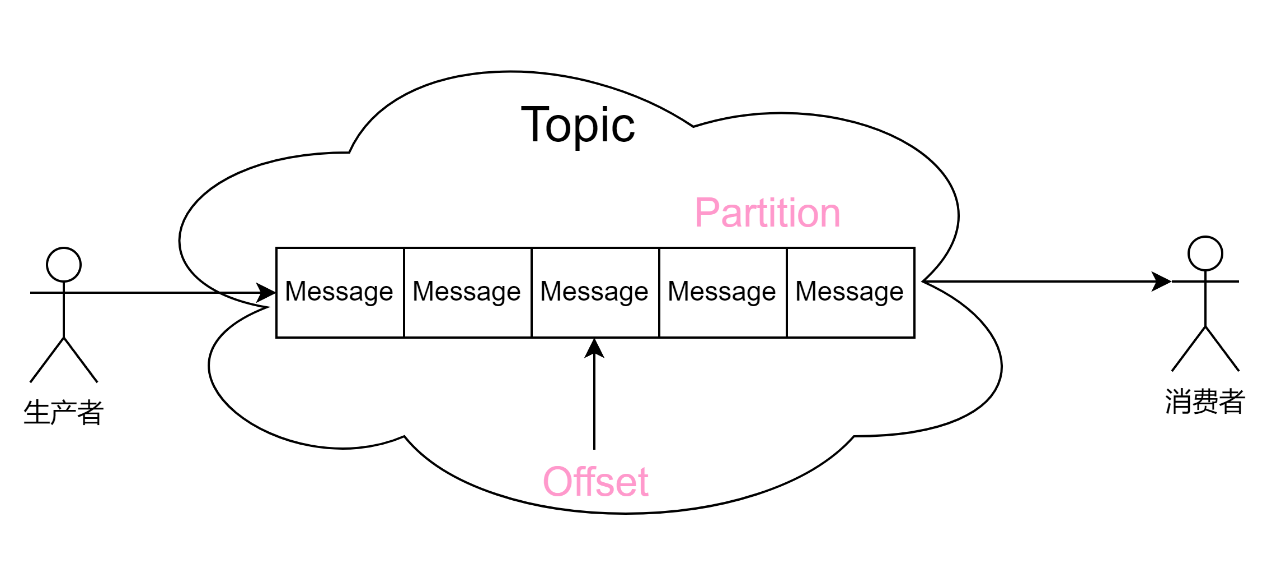
kafaka的partition是什么？

1 kafka 有 topic，生产者和消费者根据topic来确定往哪里发送消息，接受消息

2 topic下面有不同的partition，一个partition代表一个队列

3 offset 表示在partition里元素的偏移量

partition是一个数据结构



消费组用什么作用？

消费者A 属于 消费组1

消费者B 属于 消费组1

一个消息进来之后，如果消费者A消费了，消费者B就不会消费，隶属与同一个消费组的消费者同一个一个消息只会消费一次

Elasticsearch中的文档有什么特点?

1. 不同文档的字段数量和名称可以不一样。文档可以有不同结构
2. 不同文档中同名的字段，类型必须一样。相同字段必须是相同类型

AO/PO/OO分别代表什么？

面向切面

面向过程

面向对象

如何构建自己的笔记系统？

1. 改变认知：就是要开多个窗口，存放不同的信息。
2. 信息是有价值的，不同价值的信息放在不同窗口里面。

习惯开多窗口，在不同窗口间进行切换

不要怕窗口开的多，说明分类很齐全，各种笔记软件结合用，不同价值信息放在不同的窗口里面

不同软件平台，不同窗口，配合起来使用

低价值信息 – 放进流水账里

一些疑问，问题 – 放进多问为什么

单独做的笔记 – 放进CSDN

|  |  |  |  |
| --- | --- | --- | --- |
|  | 低价值信息 | 中等价值信息 | 高价值信息 |
| 放在哪里 | 流水账.md | 多问为什么.docx | CSDN |
| 内容 |  | 随记、疑惑、学习心得  以提问题的形式记录 | 单独做的笔记思考 |
| 是否同步 | 不用同步 | git同步 | 云同步 |

要对信息有一个分类，然后记录在不同的文件、软件、窗口里面，进行分流

可以不断改变这套打法，目前的打法是这样

核心：把遇到的信息进行分类归纳，放在不同的地方

关键：习惯开很多个窗口，进行信息流程，窗口多开一点，更好做各种笔记等

窗口开的越多越好，习惯多开窗口，复杂地做笔记

要习惯开很多窗口，在不同窗口进行切换，这样可以更好地进行信息流转，任务栏的窗口越多越好，窗口开了就不要随便关，就打开就是，把常用的软件固定到任务栏。

信息有价值

窗口越多越好

窗口打开了就不要关闭

任务栏软件铺满最好

要适应复杂，不要太洁净，不然不好做复杂的事情

操作习惯改变

面对复杂工程、笔记等才可以游刃有余

主动变得复杂

什么是消息？

是传输信息的载体，是生产和消费数据的最小单位

Topic、消息、生产者、消费者、生产组、消费组之间的关系？

1. 一个Topic可以有多条消息，一条消息只能属于一个Topic
2. 一个生产者可以生产多个Topic的消息
3. 一个消费者只能订阅和消费一个Topic的消息，一个Topic里的消息可以被多个不同消费组里的消费者消费

RocketMQ里的基本概念和Kafka里的基本概念总结？

RocketMQ：消息、Topic、标签、队列、分片、消息标识、消费组

Kafka： 消息、Topic、 、分区、 、 、消费组

什么是分布式锁？

在分布式系统或集群模式下，满足多进程可见并且互斥的锁

讲一下SETNX 和EXPIRE

setnx key value

expire key timevalue

如何保证setnx和expire同时成功、同时失败？

set key value nx ex timevalue

这样就可以保证原子性操作

为什么要用分布式锁？

因为普通的锁只能对同一个JVM的线程进行互斥

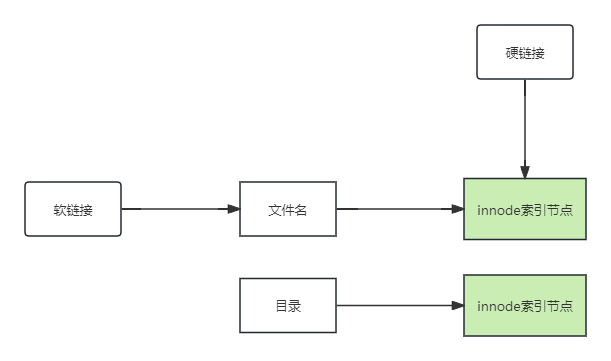
如何创建一个符号链接？

符号链接，又被称为软链接，soft link

link -symbolic source.txt linktosource.txt

为什么硬链接不能链接目录?

因为可能导致循环引用



如何查看一个日志？

tail -f

tail 会默认显示文件后10行内容

-f follow，会监视文件的变化，跟随

如果文件有新的内容，会打印在控制台

grep 参数 日志文件名

-i ignore 忽略大小写

-C context 上下文

Linux 系统中符号链接与硬链接的区别？

|  |  |  |
| --- | --- | --- |
|  | 符号链接 | 硬链接 |
| 链接类型 | 指向目标文件路径的快捷方式 | 指向目标文件innode的另一个文件名 |
| 是否指向innode | 不指向innode，指向文件路径 | 直接指向文件innode |
| 跨文件系统 | 可以 | 不可以 |
| 是否可以链接目录 | 可以 | 不可以 |
| 目标文件删除影响 | 符号链接会变成断链 | 没有影响，一个文件只有所有硬链接被删除，这个文件才会被删除 |



当Linux主机的内存不够的时候，如何使用swap增加内存？

可以写一篇

常见的Linux目录结构？

/ 根目录

/bin 用户二进制文件 ls cp命令

/sbin 系统二进制文件 system binaray fdisk systemctl命令

/etc 配置文件目录，etc.. 诸如，等等

等等，及其他（et cetera）

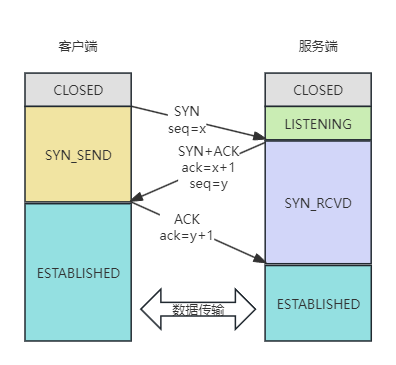
ps aux命令解释

process status all user exclude

netstat -napt命令解释？

network statistics -numerical -all -program -tcp

说一下三次握手的大致过程？



* 首先，客户端和服务端都处于CLOSED状态，服务端启动某个进程，监听某个端口，处于LISTENING状态。
* 客户端向服务端发起连接，发送SYN报文，并发送一个初始序列号ISN，客户端进入SYN\_SEND状态。
* 服务端接收到客户端发来SYN报文，同样向客户端返回SYN+ACK报文，以及服务端的初始序列号ISN，和确认号ISN+1，表示前面ISN的序列号的数据都已经正常接收。服务端进入SYNC\_RCVD状态。
* 客户端接收到服务端发来的SYN+ACK报文后，进行第三次握手，也就是最后一次握手，发送ACK确认报文，确认号为服务端发送的序列号加1。客户端进入ESTABLISHED状态，该ACK报文可以包含数据，也可以不包含数据。
* 服务端接收到客户端发送的ACK报文后，也进入ESTABLISHED状态。

ISN的英文全称？

Initial Sequential Number

序列号

为什么是三次握手而不是两次握手？

1. 阻止历史重复连接的初始化。

假设这样一个场景，客户端发送SYN报文到服务端准备建立连接，由于网络阻塞，第一次的SYN包滞留在网络中没有到达服务端，于是客户端重新发送了一个SYN报文，进行连接的初始化，但第一次发送的旧的SYN报文先于最新的SYN报文到达了服务端。如果只有两次握手，服务端接收到这个旧的SYN报文就会返回一个SYN+ACK报文，进入连接状态，但此时客户端和服务端建立连接的是一个旧的初始化连接，服务端返回的确认号并不是客户端想要的序列号，所以会造成数据混乱。如果使用三次握手，第二次握手后，服务端并不会进入连接状态，而是进入SYN\_RCVD状态，此时客户端根据上下文判断服务端发过来的确认号并不是自己想要的，所以会发送RST报文进行连接的中断。(Reset)

1. 同步双方之间的初始序列号。

如果只有两次握手，客户端发送SYN和自己的初始化序列号之后，服务端返回SYN+ACK报文，客户端的序列号被服务端同步确认，但是服务端的序列号并没有受到确认号，也就无法进行同步。

1. 避免资源浪费。

客户端发送第一次SYN报文，在网络中阻塞了，然后重新发送了一段SYN报文与服务端建立连接，双方传输完数据后通过四次挥手断开连接。此时在网络中阻塞的SYN报文到达服务端，由于只有两次握手，服务端接收到客户端的SYN报文后向返回SYN和确认报文，然后进入连接状态，但此时客户端以及关闭，所以服务端迟迟收不到客户端的回应，会一致处于连接状态，造成资源的浪费。

第三次握手可以携带数据吗？第二次握手呢？

第三次可以携带数据

如果第三次的ACK报文丢了，后面传输数据的报文，由于也携带ACK，会被服务端当作第三次握手

第一次不能携带数据的原因：防止恶意握手请求

第二次不能携带数据的原因：因为服务端通过第一次握手只确认了客户端的数据发送能力，但是还没有确认数据接收能力，所以不能携带数据。

三次握手中最后一次ACK包丢失会发生什么？

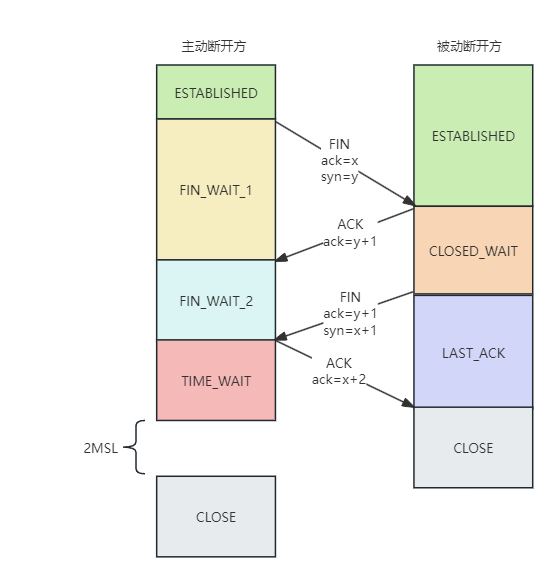
这要分情况看:

1. 后续数据包带有ACK, 服务端汇报数据包当场第三次握手。
2. 后续数据包没有ACK，服务端重新发送第二次握手的SYN+ACK报文，此时客户端会重新发送ACK报文。

为什么不是四次握手？

因为客户端向服务端发送SYN报文和ACK报文的两次握手被合并到了.

讲一下四次挥手的过程?



断开连接分为主动断开方和被动断开方，假设客户端是主动断开方，则:

* 客户端你想服务端发送FIN报文，进入FIN\_WAIT\_1状态。
* 服务端接收到FIN报文后知道客户端想要断开连接了，会返回客户端一个ACK确认报文，然后进入CLOSE\_WAIT状态，此时客户端已经没有数据要发给服务端了，但是服务端可能有一些还没有处理完的数据要发给客户端。
* 服务端处理完所有的数据后，向客户端发送FIN报文，进入LAST\_ACK状态。
* 客户端收到服务端的FIN报文后，向服务端发送ACK确认报文，服务端接收到客户端的确认报文后进入CLOSE状态。客户带进入TIME\_WAIT状态
* 客户端等待2MSL时间后进入CLOSE状态。

什么是MSL？

Maximum Segment Lifetime

报文最大生存时间

为什么等待2MSL时间?

MSL是报文在网络中的最大存货时间，从客户端发送最后一个ACK报文开始算，如果出现故障，等到服务端发送重传报文或者其它中断报文最多等待2MSL的时间。

网络中可能存在主动关闭方的数据包，这些数据包在被动关闭方接收后会响应给主动关闭方，这一来一回所以最多要等到2倍的MSL时间。

为什么需要挥手四次?

因为服务端的ACK和FIN报文不能合并到一起发送，因为在收到客户端关闭连接请求时，此时虽然从客户端到服务端已经没有数据发送，但是服务端还有数据没处理完需要发送给客户带你，所以服务端会先发给客户带一个ACK确认请求，然后等数据处理完后再发送FIN结束连接的报文。

说一下四次挥手的过程？

主动断开方和被动断开方

客户端首先向服务端发送FIN报文，进入FIN\_WAIT\_1状态

服务端向客户端返回ACK报文，进入CLOSE\_WAIT状态

客户端接收服务端的ACK报文，进入FIN\_WAIT\_2状态

服务端向客户端发送FIN报文，进入LAST\_ACK状态

客户端发送ACK报文，进入TIME\_WAIT 状态

服务端收到客户端的ACK报文，进入CLOSE状态

客户端等待2MSL时间，进入CLOSE状态

为什么需要四次呢？

因为服务端不能讲FIN和ACK合并到一起发送，当第一次握手客户端主动发起连接关闭请求时，服务端有可能还有没有处理完的数据，此时服务端只能向客户端发送ACK确认报文，只有当服务端讲数据全部处理完之后才会想客户端发送FIN报文。

TIME-WAIT 干嘛的，CLOSE\_WAIT干嘛的，在哪一个阶段？

TIME\_WAIT是主动关闭连接方，第四次握手，向被动关闭连接方发送ACK确认报文后进入的状态，此时主动关闭连接方会等待2MSL时间，保证网络中已经不存在这个连接相关的报文后进入CLOSE状态

CLOSE\_WAIT是第二次握手，被动关闭连接方向主动关闭连接方发送ACK确认报文后进入的状态，在该阶段下， 客户端到服务端的数据连接已经关闭，而服务端到客户端的连接还在，服务端还可以向客户端发送数据。

服务端主动发起关闭还是客户端主动发起关闭TCP？

都可以

主动关闭一方会有TIME\_WAIT状态，被动关闭一方会有CLOSE\_WAIT状态。

网络通信的5层模型?

物理层：两台计算机需要通信，物理层面需要用网线、光纤、电话线…等等介质连接起来，传输01电信号。物理层负责把两台计算机连接起来，然后通过高低位电频传输01这样的电信号。

数据链路层

网络层

传输层

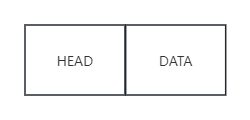
应用层

物理层：负责讲两台计算机连接起来，传输高低电频的01信号，解决两台计算机如何通信的问题

数据链路层，两台计算机通过物理层已经连接，可以传输01这样的bit流，但是这些01bit流是什么意思，如何定义一套规则来识别这些电信号，这是数据链路层要解决的问题，为了解决两台计算机之间通信的bit流识别问题，数据链路层定义了很多协议

以太网协议

帧



由帧头和数据组成

一帧的大小为64-1518个字节，如果传输的数据量很大，需要将数据切分为多个帧进行传输

一台计算机在物理层可以连接多台计算机，那么数据究竟是发给谁的，需要有一个区分的表示，这就是MAC地址

进入网络的每一个计算机都会有网卡接口，每一个网卡接口都会有一个唯一的地址，这就是MAC地址，计算机之间的数据传输，就是通过MAC地址来唯一寻址的，MAC地址一共有48位组成，在网卡生产时就被唯一标识了。

计算机A通过广播机制将数据广播到同一子网的计算机中，收到以太网帧的计算机会取出帧头里面的MAC地址，然后与自身的MAC地址做判断，如果不同则丢弃，如果相同，就会接收该数据。

ARP 地址解析协议，初始时，计算机A是不知道和自己处于同一个子网下的计算机有哪些MAC地址，于是，A会进行广播，并在目的MAC地址做特殊标识，然后同一子网下的计算机就知道需要向这个广播的源MAC地址进行回复。这样，A就获取了同一子网下的不同计算机的MAC地址，对于不同子网的计算机，在传输数据时，会有网关进行转发。

网络层，我们所处的网络是由无数个子网组成的，子网之间的数据传输需要泳道IP协议，我们如何知道哪些IP地址是属于同一个子网，通过子网掩码，将IP地址与子网掩码做and运算，如果结果相同，则说明这两台计算机处于同一个子网中。IP地址由32位组二进制数组成。通过IP地址与子网掩码做and运算，结果非零的部分就是子网。

ARP地址解析协议会广播数据，但数据里面会包含目标主机的IP地址，同一个子网的主机会把数据拿出来，并于自身IP地址做比较，如果相同，就说明A需要我回复自身的MAC地址，此时A的MAC地址我是已知的，所以广播以下，目的MAC是A的MAC地址即可。网络下处理IP地址外，还有DNS域名系统。

网络层建立了主机到主机的通信

传输层建立了端口到端口的通信，数据从计算机A发送到计算机B之后，B计算机上面运行着各种各样的程序，这些数据该给哪个进程，就需要端口号的帮助。

传输层两大协议TCP和UDP

TCP提高可靠传输，UDP提供不可靠传输

应用层，HTPP协议，STAMP协议等都是应用层的协议，用于解析和处理不同格式的数据。

计算机如何知道要把数据发给交换机还是路由器？

判断源IP和目的IP是不是同一个子网，如果是，则交给交换机

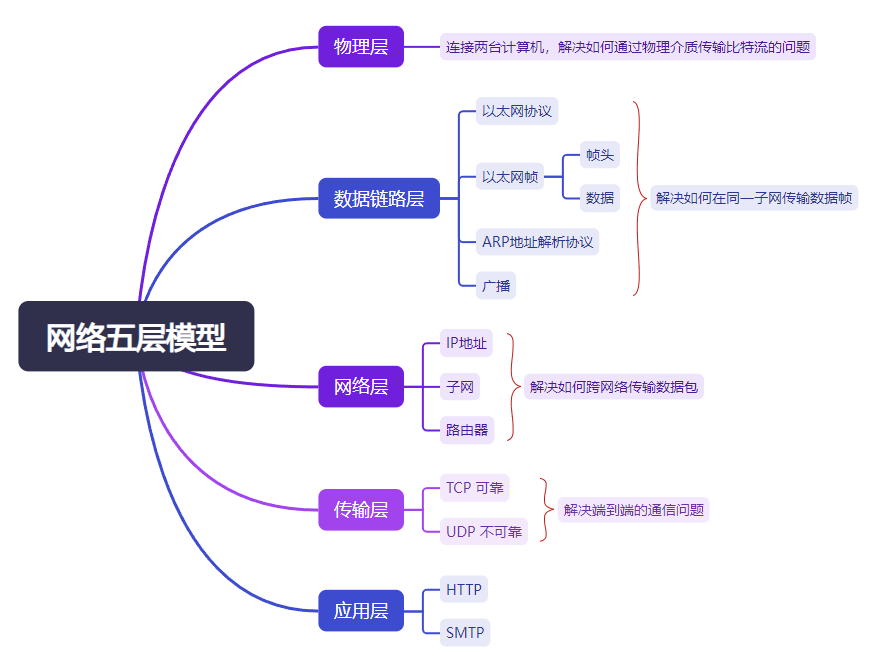
如果不是，则交给路由器

计算机A如何知道哪个设备是路由器?

发给默认网关的IP地址即可

需要在计算机A中配置这样一个网关地址

介绍一下网络五层模型？



为什么要分出四层五层这样的模型呢？

提供了一种结构化的方法来讨论整个网络协议。在实现某一层服务时，只需确保其对上层提供相应的服务，并且可以使用下层服务，至于下层服务是如何实现的，并不关心。

什么是 TCP ？举几个应用了 TCP 协议的例子？