1：es介绍

Elasticsearch是一个基于Lucene的实时的分布式搜索和分析引擎。设计用于云计算中，
能够达到实时搜索，稳定，可靠，快速，安装使用方便。基于RESTful接口。
普通请求是...get?a=1
rest请求....get/a/1

2：全文搜索的工具有哪些

Lucene Solr Elasticsearch

3：es的bulk的引用场景

1.bulk API可以帮助我们同时执行多个请求

2.create 和index的区别

如果数据存在，使用create操作失败，会提示文档已经存在，使用index则可以成功执行。

3.可以使用文件操作

使用文件的方式

vi requests

curl -XPOST/PUT localhost:9200/*bulk --data-binary @request;*

bulk请求可以在URL中声明/index 或者/*index*/type

4.bulk一次最大处理多少数据量

bulk会把将要处理的数据载入内存中，所以数据量是有限制的

最佳的数据量不是一个确定的数值，它取决于你的硬件，你的文档大小以及复杂性，你的索引以及搜索的负载

一般建议是1000-5000个文档，如果你的文档很大，可以适当减少队列,大小建议是5-15MB，默认不能超过100M，

可以在es的配置文件中修改这个值http.max_content_length: 100mb

5.版本控制的一个问题

在读数据与写数据之间如果有其他线程进行写操作，就会出问题，es使用版本控制才避免这种问题。

在修改数据的时候指定版本号，操作一次版本号加1。

6.es的两个web访问工具

BigDesk Plugin (作者 Lukáš Vlček)　　简介：监控es状态的插件,推荐！主要提供的是节点的实时状态监控，包括jvm的情况，linux的情况，elasticsearch的情况

Elasticsearch Head Plugin (作者 Ben Birch) 简介：很方便对es进行各种操作的客户端。

4：核心概念

集群 cluster**

代表一个集群，集群中有多个节点，其中有一个为主节点，这个主节点是可以通过选举产生的，主从节点是对于集群内部来说的。

es的一个概念就是去中心化，字面上理解就是无中心节点，这是对于集群外部来说的，因为从外部来看es集群，在逻辑上是个整体，

你与任何一个节点的通信和与整个es集群通信是等价的。

主节点的职责是负责管理集群状态，包括管理分片的状态和副本的状态，以及节点的发现和删除。

只需要在同一个网段之内启动多个es节点，就可以自动组成一个集群。

默认情况下es会自动发现同一网段内的节点，自动组成集群。

分片 shards

代表索引分片，es可以把一个完整的索引分成多个分片，这样的好处是可以把一个大的索引拆分成多个，分布到不同的节点上。构成分布式搜索。分片的数量只能在索引创建前指定，并且索引创建后不能更改。

可以在创建索引库的时候指定

curl -XPUT 'localhost:9200/test1/' -d'{"settings":{"number_of_shards":3}}'

默认是一个索引库有5个分片

index.number_of_shards: 5

副本 replicas*

代表索引副本，es可以给索引设置副本，副本的作用一是提高系统的容错性，当某个节点某个分片损坏或丢失时可以从副本中恢复。

二是提高es的查询效率，es会自动对搜索请求进行负载均衡。

可以在创建索引库的时候指定，副本数后期可以更改。

curl -XPUT 'localhost:9200/test2/' -d'{"settings":{"number_of_replicas":2}}'

默认是一个分片有1个副本

index.number_of_replicas: 1

数据重新分布 recovery *

代表数据恢复或叫数据重新分布，es在有节点加入或退出时会根据机器的负载对索引分片进行重新分配，挂掉的节点重新启动时也会进行数据恢复。

数据的持久化操作 gateway*

代表es索引的持久化存储方式，es默认是先把索引存放到内存中，当内存满了时再持久化到硬盘。当这个es集群关闭再重新启动时就会从gateway中读取索引数据。es支持多 种类型的gateway，有本地文件系统（默认），分布式文件系统，Hadoop的HDFS和amazon的s3云存储服务。

自动发现机制 discovery.zen*

代表es的自动发现节点机制，es是一个基于p2p的系统，它先通过广播寻找存在的节点，再通过多播协议来进行节点之间的通信，同时也支持点对点的交互。

集群或节点与客户端交互的方式 Transport*

代表es内部节点或集群与客户端的交互方式，默认内部是使用tcp协议进行交互，同时它支持http协议（json格式）、thrift、servlet、memcached、zeroMQ等的传输协议（通过插件方式集成）。

index	和rdbms中的数据库相似
type	表
document	行
field	列

5：serachType

四种搜索类型，详细介绍

分布式搜索的流程

先把查询请求发送给集群的某一个节点

某个节点把最终的结果返回给客户端

QUERY_AND_FETCH

1：客户端把请求发送给集群中的某一个节点，这个节点会把查询请求发 送给所有分片去执行，

2：每个分片会把查询的数据(包含数据的分值,以及数据的详细内容)返回给某一个节点进行汇总，排序，然后把这些数据返回给客户端

这样客户端可能会收到(10*分片数量) 的数据

这种方案，数据量和排名都有问题。

优点：效率高，查询速度快

QUERY_THEN_FETCH（默认）

1：客户端把请求发送给集群中的某一个节点，这个节点会把查询请求发送给所有分片去执行，

2：每个分片会把查询的数据(包含数据的分值,以及数据ID)返回给某一个 节点进行汇总，排序，取前10名

3：根据前10名的id到对应的分片查询数据的详细内容，返回给客户端

这种方案，解决了数据量的问题。

但是排名还有有问题。

(DFS：初始化散发过程)

DFS_QUERY_AND_FETCH

1：在查询之前，会把所有分片的词频和文档频率(打分依据)汇总到一块

2：客户端把请求发送给集群中的某一个节点，这个节点会把查询请求发送给所有分片去执行，

3：每个分片会把查询的数据(包含数据的分值,以及数据的详细内容)返回 给某一个节点进行汇总，排序，然后把这些数据返回给客户端
解决了排名的问题
还存在数据量的问题

DFS_QUERY_THEN_FETCH
1：在查询之前，会把所有分片的词频和文档频率(打分依据)汇总到一块
2：客户端把请求发送给集群中的某一个节点，这个节点会把查询请求发送给所有分片去执行，
3：每个分片会把查询的数据(包含数据的分值,以及数据ID)返回给某一个节点进行汇总，排序，取前10名
4：根据前10名的id到对应的分片查询数据的详细内容，返回给客户端
既解决了排名问题，也解决了数据量的问题
但是性能最低
总结一下，从性能考虑QUERY_AND_FETCH是最快的，DFS_QUERY_THEN_FETCH是最慢的。从搜索的准确度来说，DFS要比非DFS的准确度更高。

高亮　　补：高亮的注意事项：
高亮的内容和原始内容是分开返回的
高亮字段的内容必须在es中存储(是否存储这个属性的值必须是true)
分组：分组统计数量或者分组统计分数
删除索引库：两种方式xurl或者java api
Timeout ：

6：建立索引和查询的流程
建立索引的流程：
首先根据空白符进行分割再切分关键词，去除停用词，如果有英文全部转换为小写，对切分的关键词建立索引，每个关键词都有对应的id，还有一个倒排索引队列存储该关键词出现在文档的id，在该文档出现的次数，在该文档出现的位置
查询的流程：
首先根据空白符进行分割，再切分关键词，去除停用词，如果有英文全部转换为小写，将切分后的到的关键词和索引库进行匹配
中文分词器-IK
es官方提供的分词插件对中文分词效果不是很好，可以集成ik分词，对中文分词效果比较好
如果想根据自己的规则进行分词，可以自定义分词库，自定义分词库文件必须以.dic结尾，词库文件的编码为utf—8 without bom，一个词语一行，将自定义的文件库加入到ES_HOME/config/ik/ 目录下，修改ik的配置文件，重启生效

7：为什么使用索引工具查询快
(使用了倒排索引的技术，大致介绍一下倒排索引，还有索引库中的词都是按照顺序排列，
后期根据一个关键词查询的时候，可以利用类似折半查找的算法，查询效率非常高)
使用了倒排索引的技术，一般我们都是这样定义id 关键词，倒排索引是关键词 id正好相反，使用索引工具进行查询时，首先得到关键词，建立倒排索引表，关键词----索引列表包含该关键词所在的文档的id、在该文档中出现的次数、在该文档中出现的位置信息，这种由属性值确定记录的位置的方式成为倒排索引。还有 索引库中的词都是按照顺序排列 ，后期根据一个关键词查询的时候，

可以利用类似折半查找的算法，查询效率非常高

8.setting 与mapping 作用
settings修改索引库默认配置
Mapping,动态mapping机制：这个机制会自定识别参数值的类型，自动给这个参数设置属性
好处：操作方便，不需要提前考虑这个字段是否在es中定义过。
弊端：针对一个未知的数据，本来不应该存储的，这样也会存储了，对数据基本没有什么可控性
在实际开发中，如果数据类型已知，建议还是关闭自动mapping。
如果数据类型未知，只能使用自定mapping机制

9：分片查询方式：
es中默认的分片查询方式为随机从分片中取数据，其他的分片查询方式还有如：
local:查询操作首先在本地查找，如果本地没有再到其他节点进行查找
_primary:只在主分片中查询
_shads:按照指定的分片进行查询，这种查询方式实现了es的极速查询
（在存储数据的时候通过rooting参数可以指定将数据分配到某一个分片中，rooting参数值相同的分配到一个分片中，后期查询时可以根据rooting参数值指定到哪个分片中查找，从而实现了极速查询）
修改源码自定义从多个节点进行查询等
10：es集群的脑裂问题

es集群有可能会出现脑裂问题，原因主要有两个：
1）如果集群中节点不在同一个网段有可能是网络延迟造成的
2）如果集群中的节点在同一个网段，有可能是主节点负载太大造成的
解决方案主要有两种：
1） 把主从节点的职责分离,设置三个储备主节点， node.master=true,node.data=false
从节点只存储数据，node.master=false,node.data=true
2)增加延迟时间
将储备主节点数最小设为n/2+1个

11：优化
适当调大系统打开的最大打开文件数，默认为1024
修改配置文件调整es的jvm内存的大小，根据服务器内存的大小，一般分配60%左右，默认是256M
分片的数量最好设置为5-20个（es中一个分片最多存20G的数据，分片数在创建索引库时就指定，而且创建后不能修改，分片数最少设置为：数据量/20G个， 如果所有分片都存满数据，需要再重新建立索引库）分片设置的过多过少都会导致检索比较慢，分片数过多会导致检索时打开比较多的文件，
也会导致多台服务器之间的通信；分片数过少会导致单个分片索引过大，所以检索速度慢。
副本数多可以提升搜索能力，但是如果设置的副本数过多也会对服务器造成额外的压力，因为需要同步数据，所以设置2-3个即可
定时对索引进行优化，合并索引片段，一个索引片段的最好不要超过1G，将多个索引片段合并成一个大的索引片段时，不要太大，太大打开会很慢。
删除一条数据时不会立即删除，而是产生一个.del的文件，在检索过程中这部分数据也会参与检索，在检索过程中会判断是否删除了，如果删除了再过滤掉，这样会降低检索效率，可以定时执行curl命令进行删除或通过程序代码进行删除。
如果项目开始的时候需要批量入库大量数据，建议将副本数设为0，因为副本存在，数据要同步到副本，增加es的压力，索引完成后再将副本数修改过来，这样可以提高索引效率。
*去掉mapping中的*all域，这个虽然会给查询带来方便，但是会增加索引时间和所以尺寸
Log输出的水平默认为trace，查询超过500ms即为慢查询，就要打日志，造成cpu，内存，io负载很高，把log水平调为info或是修改配置将查询超时时间调的长一些。
12：典型的应用场景
es+hbase
利用两个框架的优点实现快速复杂查询和海量数据存储
Es+hbase：利用这两个框架的优点实现快速复杂查询和海量数据存储。
Es通过建立索引实现快速查询，它也可以存储但是不适合海量数据的存储，只存储需要那些需要从索引库中直接返回给客户的内容
Hbase适合海量数据存储，按rowkey查询可以实现快速查询，但是按列查询效率不高，所以结合es实现按字段快速查询
例如：针对海量的文章数据进行存储和快速复杂查询服务就可以通过es+hbase
比如文章数据有：（如果是面试题，需要问清楚需求，需要根据哪些字段进行查询，哪些内容直接从索引库中直接返回给客户，再进行下面的设置）
Es的设计：
Id：es内置，既建立索引也存储
Title：既建立索引也存储
Author：不建立索引，存储
Describe：既建立索引也存储

Content：建立索引不存储
Hbase的设计：
Rowkey的设计：文章的id
一个列族：info
列限定符：title，author，describe，content
13.客户端请求过程
文档能够通过主要分片(Primary Shard)或者任意一个副本分片(Replica Shard)获取。
每个步骤解释如下：
客户端(Client)发送一个请求到节点1。
该节点利用文档的_id字段来判断该文档属于分片0。分片0的分片拷贝(主要分片或者是副本分片)存在于所有的3个节点上。这一次，它将请求转发到了节点2。
节点2将文档返回给节点1，节点1随即将文档返回给客户端。 对于读请求(Read Request)，
请求节点(Requesting Node)每次都会选择一个不同的分片拷贝来实现负载均衡 -循环使用所有的分片拷贝。
可能存在这种情况，当一份文档正在被索引时，该文档在主要分片已经就绪了，但是还未被拷贝到其他副本分片上。
此时副本分片或许报告文档不存在(译注：此时有读请求来获取该文档)，然而主要分片能够成功返回需要的文档 。

一旦索引请求返回给用户的响应是成功，那么文档在主要分片以及所有副本分片上都是可用的。

==========

1.为什么要使用Elasticsearch?

因为在我们商城中的数据，将来会非常多，所以采用以往的模糊查询，模糊查询前置配置，会放弃索引，导致商品查询是全表扫面，在百万级别的数据库中，效率非常低下，而我们使用ES做一个全文索引，我们将经常查询的商品的某些字段，比如说商品名，描述、价格还有id这些字段我们放入我们索引库里，可以提高查询速度。

2.Elasticsearch是如何实现Master选举的?

Elasticsearch的选主是ZenDiscovery模块负责的，主要包含Ping（节点之间通过这个RPC来发现彼此）和Unicast（单播模块包含一个主机列表以控制哪些节点需要ping通）这两部分；

对所有可以成为master的节点（node.master: true）根据nodeId字典排序，每次选举每个节点都把自己所知道节点排一次序，然后选出第一个（第0位）节点，暂且认为它是master节点。

如果对某个节点的投票数达到一定的值（可以成为master节点数n/2+1）并且该节点自己也选举自己，那这个节点就是master。否则重新选举一直到满足上述条件。
补充：master节点的职责主要包括集群、节点和索引的管理，不负责文档级别的管理；data节点可以关闭http功能。

3.Elasticsearch中的节点（比如共20个），其中的10个选了一个master，另外10个选了另一个master，怎么办?

当集群master候选数量不小于3个时，可以通过设置最少投票通过数量（discovery.zen.minimum_master_nodes）超过所有候选节点一半以上来解决脑裂问题；
当候选数量为两个时，只能修改为唯一的一个master候选，其他作为data节点，避免脑裂问题。

4.详细描述一下Elasticsearch索引文档的过程。

协调节点默认使用文档ID参与计算（也支持通过routing），以便为路由提供合适的分片。

shard = hash(document_id) % (num_of_primary_shards)

当分片所在的节点接收到来自协调节点的请求后，会将请求写入到Memory Buffer，然后定时（默认是每隔1秒）写入到Filesystem Cache，这个从Momery Buffer到Filesystem Cache的过程就叫做refresh；

当然在某些情况下，存在Momery Buffer和Filesystem Cache的数据可能会丢失，ES是通过translog的机制来保证数据的可靠性的。其实现机制是接收到请求后，同时也会写入到translog中，当Filesystem cache中的数据写入到磁盘中时，才会清除掉，这个过程叫做flush；

在flush过程中，内存中的缓冲将被清除，内容被写入一个新段，段的fsync将创建一个新的提交点，并将内容刷新到磁盘，旧的translog将被删除并开始一个新的translog。

flush触发的时机是定时触发（默认30分钟）或者translog变得太大（默认为512M）时；

5.详细描述一下Elasticsearch更新和删除文档的过程

删除和更新也都是写操作，但是Elasticsearch中的文档是不可变的，因此不能被删除或者改动以展示其变更；

磁盘上的每个段都有一个相应的.del文件。当删除请求发送后，文档并没有真的被删除，而是在.del文件中被标记为删除。该文档依然能匹配查询，但是会在结果中被过滤掉。当段合并时，在.del文件中被标记为删除的文档将不会被写入新段。

在新的文档被创建时，Elasticsearch会为该文档指定一个版本号，当执行更新时，旧版本的文档在.del文件中被标记为删除，新版本的文档被索引到一个新段。旧版本的文档依然能匹配查询，但是会在结果中被过滤掉。

6.详细描述一下Elasticsearch搜索的过程

搜索被执行成一个两阶段过程，我们称之为 Query Then Fetch；

在初始查询阶段时，查询会广播到索引中每一个分片拷贝（主分片或者副本分片）。 每个分片在本地执行搜索并构建一个匹配文档的大小为 from + size 的优先队列。PS：在搜索的时候是会查询Filesystem Cache的，但是有部分数据还在Memory Buffer，所以搜索是近实时的。

每个分片返回各自优先队列中 所有文档的 ID 和排序值 给协调节点，它合并这些值到自己的优先队列中来产生一个全局排序后的结果列表。

接下来就是 取回阶段，协调节点辨别出哪些文档需要被取回并向相关的分片提交多个 GET 请求。每个分片加载并 丰富 文档，如果有需要的话，接着返回文档给协调节点。一旦所有的文档都被取回了，协调节点返回结果给客户端。

补充：Query Then Fetch的搜索类型在文档相关性打分的时候参考的是本分片的数据，这样在文档数量较少的时候可能不够准确，DFS Query Then Fetch增加了一个预查询的处理，询问Term和Document frequency，这个评分更准确，但是性能会变差。

9.Elasticsearch对于大数据量（上亿量级）的聚合如何实现?

Elasticsearch 提供的首个近似聚合是cardinality 度量。它提供一个字段的基数，即该字段的distinct或者unique值的数目。它是基于HLL算法的。HLL 会先对我们的输入作哈希运算，然后根据哈希运算的结果中的 bits 做概率估算从而得到基数。其特点是：可配置的精度，用来控制内存的使用（更精确 = 更多内存）；小的数据集精度是非常高的；我们可以通过配置参数，来设置去重需要的固定内存使用量。无论数千还是数十亿的唯一值，内存使用量只与你配置的精确度相关 .

10.在并发情况下，Elasticsearch如果保证读写一致?

可以通过版本号使用乐观并发控制，以确保新版本不会被旧版本覆盖，由应用层来处理具体的冲突；

另外对于写操作，一致性级别支持quorum/one/all，默认为quorum，即只有当大多数分片可用时才允许写操作。但即使大多数可用，也可能存在因为网络等原因导致写入副本失败，这样该副本被认为故障，分片将会在一个不同的节点上重建。

对于读操作，可以设置replication为sync(默认)，这使得操作在主分片和副本分片都完成后才会返回；如果设置replication为async时，也可以通过设置搜索请求参数_preference为primary来查询主分片，确保文档是最新版本。

14.ElasticSearch中的集群、节点、索引、文档、类型是什么?

群集是一个或多个节点（服务器）的集合，它们共同保存您的整个数据，并提供跨所有节点的联合索引和搜索功能。群集由唯一名称标识，默认情况下为“elasticsearch”。此名称很重要，因为如果节点设置为按名称加入群集，则该节点只能是群集的一部分。

节点是属于集群一部分的单个服务器。它存储数据并参与群集索引和搜索功能。

索引就像关系数据库中的“数据库”。它有一个定义多种类型的映射。索引是逻辑名称空间，映射到一个或多个主分片，并且可以有零个或多个副本分片。 MySQL =>数据库 ElasticSearch =>索引

文档类似于关系数据库中的一行。不同之处在于索引中的每个文档可以具有不同的结构（字段），但是对于通用字段应该具有相同的数据类型。 MySQL => Databases => Tables => Columns / Rows ElasticSearch => Indices => Types =>具有属性的文档

类型是索引的逻辑类别/分区，其语义完全取决于用户。

15.ElasticSearch中的分片是什么?

在大多数环境中，每个节点都在单独的盒子或虚拟机上运行。

索引 - 在Elasticsearch中，索引是文档的集合。

分片 -因为Elasticsearch是一个分布式搜索引擎，所以索引通常被分割成分布在多个节点上的被称为分片的元素。

问题一：

什么是ElasticSearch?

Elasticsearch是一个基于Lucene的搜索引擎。它提供了具有HTTP Web界面和无架构JSON文档的分布式，多租户能力的全文搜索引擎。Elasticsearch是用Java开发的，根据Apache许可条款作为开源发布。

问题三：

Elasticsearch中的倒排索引是什么?

倒排索引是搜索引擎的核心。搜索引擎的主要目标是在查找发生搜索条件的文档时提供快速搜索。倒排索引是一种像数据结构一样的散列图，可将用户从单词导向文档或网页。它是搜索引擎的核心。其主要目标是快速搜索从数百万文件中查找数据。

问题四：

ElasticSearch中的集群、节点、索引、文档、类型是什么?

- 群集是一个或多个节点（服务器）的集合，它们共同保存您的整个数据，并提供跨所有节点的联合索引和搜索功能。群集由唯一名称标识，默认情况下为“elasticsearch”。此名称很重要，因为如果节点设置为按名称加入群集，则该节点只能是群集的一部分。
- 节点是属于集群一部分的单个服务器。它存储数据并参与群集索引和搜索功能。

- 索引就像关系数据库中的“数据库”。它有一个定义多种类型的映射。索引是逻辑名称空间，映射到一个或多个主分片，并且可以有零个或多个副本分片。 MySQL =>数据库 ElasticSearch =>索引
- 文档类似于关系数据库中的一行。不同之处在于索引中的每个文档可以具有不同的结构（字段），但是对于通用字段应该具有相同的数据类型。 MySQL => Databases => Tables => Columns / Rows ElasticSearch => Indices => Types =>具有属性的文档
- 类型是索引的逻辑类别/分区，其语义完全取决于用户。

问题五：

ElasticSearch是否有架构?

ElasticSearch可以有一个架构。架构是描述文档类型以及如何处理文档的不同字段的一个或多个字段的描述。Elasticsearch中的架构是一种映射，它描述了JSON文档中的字段及其数据类型，以及它们应该如何在Lucene索引中进行索引。因此，在Elasticsearch术语中，我们通常将此模式称为“映射”。

Elasticsearch具有架构灵活的能力，这意味着可以在不明确提供架构的情况下索引文档。如果未指定映射，则默认情况下，Elasticsearch会在索引期间检测文档中的新字段时动态生成一个映射。

问题六：

ElasticSearch中的分片是什么?

在大多数环境中，每个节点都在单独的盒子或虚拟机上运行。

- 索引 - 在Elasticsearch中，索引是文档的集合。
- 分片 -因为Elasticsearch是一个分布式搜索引擎，所以索引通常被分割成分布在多个节点上的被称为分片的元素。

问题七：

ElasticSearch中的副本是什么?

一个索引被分解成碎片以便于分发和扩展。副本是分片的副本。一个节点是一个属于一个集群的ElasticSearch的运行实例。一个集群由一个或多个共享相同集群名称的节点组成。

问题八：

ElasticSearch中的分析器是什么?

在ElasticSearch中索引数据时，数据由为索引定义的Analyzer在内部进行转换。 分析器由一个Tokenizer和零个或多个TokenFilter组成。编译器可以在一个或多个CharFilter之前。分析模块允许您在逻辑名称下注册分析器，然后可以在映射定义或某些API中引用它们。

Elasticsearch附带了许多可以随时使用的预建分析器。或者，您可以组合内置的字符过滤器，编译器和过滤器器来创建自定义分析器。

问题九：

什么是ElasticSearch中的编译器?

编译器用于将字符串分解为术语或标记流。一个简单的编译器可能会将字符串拆分为任何遇到空格或标点的地方。Elasticsearch有许多内置标记器，可用于构建自定义分析器。

问题十一：

启用属性，索引和存储的用途是什么？

enabled属性适用于各类ElasticSearch特定/创建领域，如index和size。用户提供的字段没有“已启用”属性。 存储意味着数据由Lucene存储，如果询问，将返回这些数据。

存储字段不一定是可搜索的。默认情况下，字段不存储，但源文件是完整的。因为您希望使用默认值(这是有意义的)，所以不要设置store属性 该指数属性用于搜索。

索引属性只能用于搜索。只有索引域可以进行搜索。差异的原因是在分析期间对索引字段进行了转换，因此如果需要的话，您不能检索原始数据。