配置详解：

ElasticSearch包含模块由：

集群管理Cluster、分片管理Shards、副本管理Replicas、恢复管理Recovery、数据源管理River、持久化存储管理Gateway、节点自动发现管理Discovery.zen、交互模块Transport。

***Cluster***

**# ---------------------------------- Cluster -----------------------------------**

**#**

**# Use a descriptive name for your cluster: //为集群起一个有实际描述意义的名字**

**#**

**#cluster.name: my-application**

**#**

**--------------------------------------------------------------------------------**

代表一个集群，集群中有多个节点，其中有一个为主节点（相应的就有一个或者多个从节点），这个主节点是可以通过选举（选择算法，类似redis的哨兵来选择master）产生的，主从节点是对于集群内部来说的（外部来看集群是一个整体，请求任何一个节点的效果与都是一样的）。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（物理分散但逻辑上是一体），你与任何一个节点的通信和与整个es集群通信是等价的。

***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（内部实现了这样一种算法来实现索引分片，所有在使用的时候不能更改，否则就不能命中正确索引）。

***Replicas***

代表索引副本，es可以设置多个索引的副本，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（基于某种负载均衡算法）。

***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。

***River***

代表es的一个数据源，也是其它存储方式（如：数据库）同步数据到es的一个方法。它是以插件方式存在的一个es服务，通过读取river中的数据并把它索引到es中，官方的river有couchDB的，RabbitMQ的，Twitter的，Wikipedia的，river这个功能将会在后面的文件中重点说到。

***Gateway***

# ---------------------------------- Gateway -----------------------------------

#

# Block initial recovery after a full cluster restart until N nodes are started:当一个集群重启的时候需要至少多少个节点启动才能执行数据恢复初始化，即在集群中开启的节点没有达到这个值的时候集群master不会进行数据的回复工作。

#

#gateway.recover\_after\_nodes: 3

#

# For more information, consult the gateway module documentation.

--------------------------------------------

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。

***discovery.zen***

# --------------------------------- Discovery ----------------------------------

#

# Pass an initial list of hosts to perform discovery when new node is started:当新的节点启动的时候，传递一个host列表以便集群中个节点之间的相互发现

# The default list of hosts is ["127.0.0.1", "[::1]"] //默认的配置是本机

#

#discovery.zen.ping.unicast.hosts: ["host1", "host2"]

#

# Prevent the "split brain" by configuring the majority of nodes (total number of master-eligible nodes / 2 + 1): 通过配置节点数来阻止脑裂（合法的master的节点数除以2+1）

#

#discovery.zen.minimum\_master\_nodes: 3

#

# For more information, consult the zen discovery module documentation.

代表es的自动发现节点机制，es是一个基于p2p（point-to-point点对点）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。

**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）。

***提示：ES的配置文件位于config/elasticsearch.yml***

***集群配置***

es启动时会严格检查各项的配置是否合适，即使将索引的配置放在了node的配置中也是不正确的，也会在启动es的时候触发异常。

# 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。

# cluster.name可以确定你的集群名称,当你的elasticsearch集群在同一个网段中elasticsearch会自动的找到具有相同cluster.name的elasticsearch服务.所以当同一个网段具有多个elasticsearch集群时cluster.name就成为同一个集群的标识.

cluster.name: elasticsearch

***节点配置-Node***

***名称***

# 节点名称同理,可自动生成（从已有的0-128中随机分配）也可手动配置.

node.name: "Franz Kafka"

***主从配置***

# 允许一个节点是否可以成为一个master节点,es是默认集群中的第一台机器为master（即第一台启动的es）,如果这台机器停止就会重新选举master.

node.master: true

# 允许该节点存储数据(默认开启)

node.data: true

# 配置文件中给出了三种配置高性能集群拓扑结构的模式,如下：

# 1. 如果你想让节点从不选举为主节点,只用来存储数据,可作为负载器

node.master: false

node.data: true

# 2. 如果想让节点成为主节点,且不存储任何数据,并保有空闲资源,可作为协调器

node.master: true

node.data: false

# 3. 如果想让节点既不称为主节点,又不成为数据节点,那么可将他作为搜索器,从节点中获取数据,生成搜索结果等

node.master: false

node.data: false

# 监控集群状态有一下插件和API可以使用:

# 查看集群健康情况

http://localhost:9200/\_cluster/health

http://localhost:9200/\_nodes

# 或者通过GUI工具查看例如：

<http://www.elasticsearch.org/overview/marvel/

http://github.com/karmi/elasticsearch-paramedic

http://github.com/lukas-vlcek/bigdesk

http://mobz.github.com/elasticsearch-head

***索引配置-Index***

# 设置索引的分片数,默认为5

index.number\_of\_shards: 5

# 设置索引的副本数,默认为1:

index.number\_of\_replicas: 1

# 配置文件中提到的最佳实践是,如果服务器够多,可以将分片提高,尽量将数据平均分布到大集群中去同时,如果增加副本数量可以有效的提高搜索性能

# 需要注意的是,"number\_of\_shards" 是索引创建后一次生成的,后续不可更改设置

# "number\_of\_replicas" 是可以通过API去实时修改设置的

**路径配置-Paths**

# 配置文件存储位置

path.conf: /path/to/conf

# 数据存储位置(单个目录设置)

path.data: /path/to/data

# 多个数据存储位置,有利于性能提升

path.data: /path/to/data1,/path/to/data2

# 临时文件的路径

path.work: /path/to/work

# 日志文件的路径

path.logs: /path/to/logs

# 插件安装路径

path.plugins: /path/to/plugins

***插件配置***

# 设置插件作为启动条件,如果一个插件没有安装,则该节点服务不会启动

#plugin.mandatory: mapper-attachments,lang-groovy

***内存配置***

# 当JVM开始写入交换空间时（swapping）ElasticSearch性能会低下,你应该保证它不会写入交换空间，设置这个属性为true来锁定内存,同时也要允许elasticsearch的进程可以锁住内存,linux下可以通过 `ulimit -l unlimited` 命令

bootstrap.memory\_lock: true

# 确保 ES\_MIN\_MEM 和 ES\_MAX\_MEM 环境变量设置为相同的值,以及机器有足够的内存分配给Elasticsearch注意:内存也不是越大越好,一般64位机器,最大分配内存别才超过32G

***网络配置***

# 设置绑定的ip地址,可以是ipv4或ipv6的,默认为0.0.0.0

network.bind\_host: 192.168.0.1

# 设置其它节点和该节点交互的ip地址,如果不设置它会自动设置,值必须是个真实的ip地址

network.publish\_host: 192.168.0.1

# 同时设置bind\_host和publish\_host上面两个参数

network.host: 192.168.0.1

# 设置节点间交互的tcp端口,默认是9300

transport.tcp.port: 9300

# 设置是否压缩tcp传输时的数据，默认为false,不压缩

transport.tcp.compress: true

# 设置对外服务的http端口,默认为9200

http.port: 9200

# 设置请求内容的最大容量,默认100mb

http.max\_content\_length: 100mb

# 使用http协议对外提供服务,默认为true,开启

http.enabled: false

***网关配置***

# gateway的类型,默认为local即为本地文件系统,可以设置为本地文件系统

gateway.type: local

# 下面的配置控制怎样以及何时启动一整个集群重启的初始化恢复过程

# (当使用shard gateway时,是为了尽可能的重用local data(本地数据))

# 一个集群中的N个节点启动后,才允许进行恢复处理

gateway.recover\_after\_nodes: 1

# 设置初始化恢复过程的超时时间,超时时间从上一个配置中配置的N个节点启动后算起

gateway.recover\_after\_time: 5m

# 设置这个集群中期望有多少个节点.一旦这N个节点启动(并且recover\_after\_nodes也符合),

# 立即开始恢复过程(不等待recover\_after\_time超时)

gateway.expected\_nodes: 2

***恢复配置***

# 下面这些配置允许在初始化恢复,副本分配,再平衡,或者添加和删除节点时控制节点间的分片分配

# 设置一个节点的并行恢复数

# 1.初始化数据恢复时,并发恢复线程的个数,默认为4

cluster.routing.allocation.node\_initial\_primaries\_recoveries: 4

# 2.添加删除节点或负载均衡时并发恢复线程的个数,默认为2

cluster.routing.allocation.node\_concurrent\_recoveries: 2

# 设置恢复时的吞吐量(例如:100mb,默认为0无限制.如果机器还有其他业务在跑的话还是限制一下的好)

indices.recovery.max\_bytes\_per\_sec: 20mb

# 设置来限制从其它分片恢复数据时最大同时打开并发流的个数,默认为5

indices.recovery.concurrent\_streams: 5

# 注意: 合理的设置以上参数能有效的提高集群节点的数据恢复以及初始化速度

***集群发现配置***

# 设置这个参数来保证集群中的节点可以知道其它N个有master资格的节点.默认为1,对于大的集群来说,可以设置大一点的值(2-4)

discovery.zen.minimum\_master\_nodes: 1

# 探查的超时时间,默认3秒,提高一点以应对网络不好的时候,防止脑裂

discovery.zen.ping.timeout: 3s

# For more information, see

# http://elasticsearch.org/guide/en/elasticsearch/reference/current/modules-discovery-zen.html

# 设置是否打开多播发现节点.默认是true.

# 当多播不可用或者集群跨网段的时候集群通信还是用单播吧

discovery.zen.ping.multicast.enabled: false

# 这是一个集群中的主节点（master）的初始列表,当节点(主节点或者数据节点)启动时使用这个列表进行探测

discovery.zen.ping.unicast.hosts: ["host1", "host2:port"]

# Slow Log部分与GC log部分略,不过可以通过相关日志优化搜索查询速度

***缓存配置***

# Cache部分:es有很多种方式来缓存其内部与索引有关的数据.其中包括filter cache

# filter cache部分:

# filter cache是用来缓存filters的结果的.默认的cache type是node type.node type的机制是所有的索引内部的分片共享filter cache.node type采用的方式是LRU方式.即:当缓存达到了某个临界值之后，es会将最近没有使用的数据清除出filter cache.让新的数据进入es.这个临界值的设置方法如下：indices.cache.filter.size 值类型：eg.:512mb 20%。默认的值是10%。

# out of memory错误避免过于频繁的查询时集群假死

# 1.设置es的缓存类型为Soft Reference,它的主要特点是据有较强的引用功能.只有当内存不够的时候,才进行回收这类内存,因此在内存足够的时候,它们通常不被回收.另外,这些引用对象还能保证在Java抛出OutOfMemory异常之前,被设置为null.它可以用于实现一些常用图片的缓存,实现Cache的功能,保证最大限度的使用内存而不引起OutOfMemory.在es的配置文件加上index.cache.field.type: soft即可.

# 2.设置es最大缓存数据条数和缓存失效时间,通过设置index.cache.field.max\_size: 50000来把缓存field的最大值设置为50000,设置index.cache.field.expire: 10m把过期时间设置成10分钟.

#index.cache.field.max\_size: 50000

#index.cache.field.expire: 10m

#index.cache.field.type: soft

# field data部分&&circuit breaker部分：

# 用于field data 缓存的内存数量,主要用于当使用排序,faceting操作时,elasticsearch会将一些热点数据加载到内存中来提供给客户端访问,但是这种缓存是比较珍贵的,所以对它进行合理的设置.

# 可以使用值：eg:50mb 或者 30％(节点 node heap内存量),默认是：unbounded

#indices.fielddata.cache.size： unbounded

# field的超时时间.默认是-1,可以设置的值类型: 5m

#indices.fielddata.cache.expire: -1

# circuit breaker部分:

# 断路器是elasticsearch为了防止内存溢出的一种操作,每一种circuit breaker都可以指定一个内存界限触发此操作,这种circuit breaker的设定有一个最高级别的设定:indices.breaker.total.limit 默认值是JVM heap的70%.当内存达到这个数量的时候会触发内存回收

# 另外还有两组子设置：

#indices.breaker.fielddata.limit:当系统发现fielddata的数量达到一定数量时会触发内存回收.默认值是JVM heap的70%

#indices.breaker.fielddata.overhead:在系统要加载fielddata时会进行预先估计,当系统发现要加载进内存的值超过limit \* overhead时会进行进行内存回收.默认是1.03

#indices.breaker.request.limit:这种断路器是elasticsearch为了防止OOM(内存溢出),在每次请求数据时设定了一个固定的内存数量.默认值是40%

#indices.breaker.request.overhead:同上,也是elasticsearch在发送请求时设定的一个预估系数,用来防止内存溢出.默认值是1

***Translog配置***

# Translog部分:

# 每一个分片(shard)都有一个transaction log或者是与它有关的预写日志,(write log),在es进行索引(index)或者删除(delete)操作时会将没有提交的数据记录在translog之中,当进行flush 操作的时候会将tranlog中的数据发送给Lucene进行相关的操作.一次flush操作的发生基于如下的几个配置

#index.translog.flush\_threshold\_ops:当发生多少次操作时进行一次flush.默认是 unlimited

#index.translog.flush\_threshold\_size:当translog的大小达到此值时会进行一次flush操作.默认是512mb

#index.translog.flush\_threshold\_period:在指定的时间间隔内如果没有进行flush操作,会进行一次强制flush操作.默认是30m

#index.translog.interval:多少时间间隔内会检查一次translog,来进行一次flush操作.es会随机的在这个值到这个值的2倍大小之间进行一次操作,默认是5s

#index.gateway.local.sync:多少时间进行一次的写磁盘操作,默认是5s

# 以上的translog配置都可以通过API进行动态的设置

elasticsearch 5.3.0 配置文件

# ======================== Elasticsearch Configuration =========================

#

# NOTE: Elasticsearch comes with reasonable defaults for most settings.

# Before you set out to tweak and tune the configuration, make sure you

# understand what are you trying to accomplish and the consequences.

#

# The primary way of configuring a node is via this file. This template lists

# the most important settings you may want to configure for a production cluster.

#

# Please consult the documentation for further information on configuration options:

# https://www.elastic.co/guide/en/elasticsearch/reference/index.html

#

# ---------------------------------- Cluster -----------------------------------

#

# Use a descriptive name for your cluster:

#

#cluster.name: my-application

#

# ------------------------------------ Node ------------------------------------

#

# Use a descriptive name for the node:

#

#node.name: node-1

#

# Add custom attributes to the node:

#

#node.attr.rack: r1

#

# ----------------------------------- Paths ------------------------------------

#

# Path to directory where to store the data (separate multiple locations by comma):

#

#path.data: /path/to/data

#

# Path to log files:

#

#path.logs: /path/to/logs

#

# ----------------------------------- Memory -----------------------------------

#

# Lock the memory on startup:

#

#bootstrap.memory\_lock: true

#

# Make sure that the heap size is set to about half the memory available

# on the system and that the owner of the process is allowed to use this

# limit.

#

# Elasticsearch performs poorly when the system is swapping the memory.

#

# ---------------------------------- Network -----------------------------------

#

# Set the bind address to a specific IP (IPv4 or IPv6):

#

#network.host: 192.168.0.1

#

# Set a custom port for HTTP:

#

#http.port: 9200

#

# For more information, consult the network module documentation.

#

# --------------------------------- Discovery ----------------------------------

#

# Pass an initial list of hosts to perform discovery when new node is started:

# The default list of hosts is ["127.0.0.1", "[::1]"]

#

#discovery.zen.ping.unicast.hosts: ["host1", "host2"]

#

# Prevent the "split brain" by configuring the majority of nodes (total number of master-eligible nodes / 2 + 1):

#

#discovery.zen.minimum\_master\_nodes: 3

#

# For more information, consult the zen discovery module documentation.

#

# ---------------------------------- Gateway -----------------------------------

#

# Block initial recovery after a full cluster restart until N nodes are started:

#

#gateway.recover\_after\_nodes: 3

#

# For more information, consult the gateway module documentation.

#

# ---------------------------------- Various -----------------------------------

#

# Require explicit names when deleting indices:

#

#action.destructive\_requires\_name: true