Node

任何时候你启动一个弹性搜索的实例，你就启动一个节点。 连接节点的集合称为cluster(超连接)。
如果您正在运行es的单个节点，那么您有一个节点的集群。

默认情况下，集群中的每个节点都可以处理HTTP和Transport 。 传输层只用于节点和Java `TransportClient`之间的通信;
HTTP层仅由外部REST客户端使用。

所有节点都知道集群中的所有其他节点，并将客户端请求转发到适当的节点。 除此之外，每个节点都有一个或多个目的：

Master-eligible node

一个节点的node.master设置为true（默认），这使得它有资格被选为主节点（超连接），它控制集群。

Data node

node.data设置为true（默认）的节点。 数据节点保存数据并执行数据相关操作，Eg: CRUD，搜索和聚合。

Ingest node

node.ingest设置为true（默认）的节点。 获取节点能够将 ingest pipeline 应用于文档，以便在索引之前变换和丰富文档。
使用沉重的请求负载，使用专门的请求节点并将主节点和数据节点标记为node.ingest：false是有意义的。

Tribe node

通过tribe*设置配置的族节点是一种特殊类型的协调节点，可以连接到多个集群，并在所有连接的集群中执行搜索和其他操作。

默认情况下，节点是主节点和数据节点，加上它可以通过请求管道预处理文档。
这对于小型集群来说非常方便，但是随着集群的发展，考虑将独立主节点与独立数据节点分开是很重要的。

协调Node

搜索请求或批量索引请求的可能涉及保存在不同数据节点上的数据。
Eg，搜索请求在由接收客户端请求的节点（协调节点）协调的两个阶段中执行。在分散阶段，协调节点将请求转发到保存数据的数据节点。
每个数据节点在本地执行请求，并将其结果返回给协调节点。
在收集阶段，协调节点将每个数据节点的结果减少为单个全局结果集。每个节点都是隐式协调节点。
这意味着将所有三个node.master，node.data和node.ingest设置为false的节点将仅作为协调节点，不能禁用。
因此，这样一个节点需要有足够的内存和CPU才能处理收集阶段。

Master-eligible node

主节点负责轻量级的集群范围的操作，Eg创建或删除索引，跟踪哪些节点是集群的一部分，以及决定哪些分片分配给哪些节点。
群集健康对于拥有稳定的主节点很重要。任何主节点（默认情况下的所有节点）可以通过选举过程被选为成为主节点(master election process
)。

!!! 主节点必须具有对数据/目录的访问（就像数据节点一样），因为节点重新启动之间的集群状态是持久存在的。

索引和搜索数据是CPU，内存和I / O密集型工作，可能会对节点资源造成压力。
为了确保您的主节点稳定而不受压力，在较大的集群中，将独立出主节点与独立出数据节点架构是很nice。尽管主节点也可以作为协调节点，
并将搜索和索引请求从客户端路由到数据节点，但最好不要为此目的使用独立主节点。
对于master节点尽可能少的工作，集群的稳定性很重要。（不想半夜 被经理喊起来 说es出问题了 你看看 那时还在和女票么么哒 那就爽了
谨记！！！ 么么哒记得关机）

要创建独立的主节点节点，请设置：

node.master: true （1） node.master角色默认启用。

node.data: false （2） 禁用node.data角色（默认启用）。

node.ingest: false （3） 禁用node.ingest角色（默认情况下启用）。

为了防止数据丢失，至关重要的是配置discovery.zen.minimum_master_nodes设置（默认为1），以便每个符合主要条件的节点都知道为了
形成集群而必须可见的最少数量的主节点。假设你有一个由两个主节点组成的集群。网络故障会破坏这两个节点之间的通信。每个节点都看到
一个主要合格的节点...本身。将minimum_master_nodes设置为默认值为1，这足以形成集群。每个节点选择自己作为新的主人（认为另一个
符合资格的节点已经死亡），结果是两个群集或一个裂脑(裂脑最痛苦了 最开始要es 这里坑了我一下
)。这两个节点永远不会重新加入，直到重新启动一个节点。已经写入重新启动的节点的任何数据都将丢失。我们可以想象一下，您有一个具有
三个主节点的节点，而minimum_master_nodes设置为2.如果一个网络分裂将一个节点与其他两个节点分离，则具有一个节点的一侧无法看到
足够的主节点，并且将意识到不能选择自己作为主人。具有两个节点的一侧将选择一个新的主控（如果需要）并继续正常工作。一旦解决了网
络拆分，单个节点将重新加入集群，并再次开始投放请求。

此设置应设置为合格的主节点的法定数量：

(master_eligible_nodes / 2) + 1

换句话说，如果有三个主节点，则最小主节点应设置为（3/2）+ 1或2：（默认是1）

也可以使用集群动态设置API在实时集群上动态更改此设置：

```
PUT _cluster/settings
{
"transient": {
"discovery.zen.minimum_master_nodes": 2
}
}
```

注解：在es集群节点之间分割主节点和数据节点的优点是，您只能拥有三个主节点节点，并将minimum_master_nodes设置为2.您无需更改此设置，无论添加到集群中的独立的数据节点数量多少。

Data nodes

数据节点保存包含已编制索引的文档的分片。 数据节点处理与CRUD，搜索和聚合相关的数据相关操作。 这些操作是I / O，内存和CPU密集型。
如果监视这些资源并重新加载更多的数据节点，这是非常重要的。拥有独立数据节点的主要优点是主控和数据角色的分离。

要创建独立数据节点，请设置：

node.master: false 禁用node.master角色（默认启用）。

node.data: true node.data角色默认启用。
node.ingest: false 禁用node.ingest角色（默认情况下启用）。

Ingest nodes

获取节点可以执行由一个或多个请求处理组成的预处理数据流。
根据请求处理器执行的操作类型和所需的资源，具有专门的请求节点是有意义的，只能执行此特定任务。

要创建一个专门的Ingest nodes，请设置：

node.master: false 禁用node.master角色（默认启用）。

node.data: false 禁用node.data角色（默认启用）。
node.ingest: true node.ingest角色默认启用。
search.remote.connect: false 禁用跨群集搜索（默认情况下启用）。

协调一个node

如果您独立了能够处理主要职责，掌握数据和预处理文档的能力，那么您将留下只能路由请求的协调节点，处理搜索减少阶段并分发批量索引。 基本上，仅协调节点表现为智能负载均衡器。仅协调节点可以通过从数据和主节点节点卸载协调节点角色来受益于大型集群。
他们加入集群并接收完整的集群状态，就像每个其他节点一样，它们使用cluster state将请求直接路由到适当的位置。将多个仅协调的节点添加到群集可能会增加整个群集的负担，因为选定的主节点必须等待每个节点的群集状态更新的确认！

要创建一个专用的协调节点，请设置：

node.master: false 禁用node.master角色（默认启用）。

node.data: false 禁用node.data角色（默认启用）。
node.ingest: false 禁用node.ingest角色（默认情况下启用）。
search.remote.connect: false 禁用跨群集搜索（默认情况下启用）。

path.data

每个数据和主资源节点都需要访问数据目录，其中将存储分片和索引和集群元数据。 path.data默认为$ ES_HOME / data，但可以在elasticsearch.yml配置文件中配置绝对路径或相对于$ ES_HOME的路径，如下所示：

path.data: /var/elasticsearch/data

类似所有节点设置一样，它也可以在启动命令行中指定为：

./bin/elasticsearch -Epath.data=/var/elasticsearch/data

当使用.zip或.tar.gz发行版时，应该将path.data设置配置为在Elasticsearch主目录之外找到数据目录，以便删除主目录而不删除数据！ RPM和Debian发行版已经为您做了这个。

node.max_local_storage_nodesedit
数据路径可以由多个节点共享，即使是来自不同群集的节点。 这对于在开发机器上测试故障切换和不同配置非常有用。
但是，在生产中，建议每个服务器只运行一个弹性节点。

默认情况下，Elasticsearch配置为防止多个节点共享相同的数据路径。
要允许多个节点（例如，在您的开发机器上），请使用设置node.max_local_storage_nodes并将其设置为大于1的正整数。

谨记 从同一数据目录运行不同的节点类型（即主节点，数据）。 这可能导致意外的数据丢失。

其他节点设置
更多的节点设置可以在Modules中找到。 特别要注意的是cluster.name，node.name和网络设置。