第 8 章

# 集群管理

8

上一章介绍了Elasticsearch中节点发现如何工作，如何调优，恢复和时光之门模块，如何通过调整某些Elasticsearch内部构件来准备高索引和高查询的集群及其内部构件。最后，使用索引模板和动态映射来轻松地控制动态索引的结构。在本章，你将了解以下内容：

- 使用Elasticsearch的快照功能；
- 使用Elasticsearch API监控集群；
- 调整集群的再平衡，以配合我们的需要；
- 使用Elasticsearch API移动分片；
- 预热；
- 使用别名来减轻每天的工作；
- 安装Elasticsearch插件；
- 使用Elasticsearch的更新设置API。

## 8.1 Elasticsearch 时光机

一个好的软件可以管理硬件故障或人为错误等异常情况。尽管几台服务器的集群很少暴露硬件问题，但不好的事情仍会发生。假设你需要恢复索引。一个可能的解决方案是取得以SQL数据库存储的所有主数据，并重新生成索引。但如果时间过长，或者更糟的是，数据只存储在Elasticsearch中，怎么办？Elasticsearch 1.0之前，创建索引的备份很不容易，过程包括关闭集群、复制数据文件等。幸好，现在可以使用快照。来看看它是如何工作的。

### 8.1.1 创建快照存储库

快照保存它创建的时间点上所有跟集群相关的数据，包括集群状态和索引的信息。至少在创建第一个快照之前，必须创建一个快照存储库。

每个存储库由名称区分，应该定义如下几个方面。

- `name`：这是存储库的唯一名称，后面会用到它。

- type：这是存储库的类型，可能的值包括fs（共享文件系统中的存储库）、url（一个通过URL访问的只读存储库）。
- settings：这是不同存储库类型需要的额外信息。

现在，创建一个文件系统存储库。请注意，集群中的每个节点都应该能访问这个目录。为创建一个新的文件系统存储库，可以执行下列命令：

```
curl -XPUT localhost:9200/_snapshot/backup -d '{
  "type": "fs",
  "settings": {
    "location": "/tmp/es_backup_folder/cluster1"
  }
}'
```

前面的命令创建一个名为backup的库，将备份文件存储在由location属性指定的目录中。Elasticsearch返回下列信息：

```
{"acknowledged":true}
```

与此同时，本地文件系统中新建了backup_folder目录，但还没有任何内容。

我们说过，第二种存储库类型是url。它需要一个url参数，而不是location，该参数指向存储库所在的地址，例如，HTTP地址。你还可以把快照存储在Amazon S3或HDFS中，使用额外的可用插件（参见https://github.com/elasticsearch/elasticsearch-cloud-aws#s3-repository 和 https://github.com/elasticsearch/elasticsearch-hadoop/tree/master/repository-hdfs）。

现在，我们有了第一个存储库，可以使用以下命令看到它的定义：

```
curl -XGET localhost:9200/_snapshot/backup?pretty
```

还可以运行以下命令检查所有存储库：

```
curl -XGET localhost:9200/_snapshot/_all?pretty
```

如果你想删除一个快照存储库，标准的DELETE命令可以帮忙：

```
curl -XDELETE localhost:9200/_snapshot/backup?pretty
```

## 8.1.2 创建快照

默认情况下，创建快照时，Elasticsearch取得所有的索引和集群设置（除了瞬时的那些）。你可以创建任意数量的快照，每个快照将持有从创建的时间点开始的所有信息。快照的创建用了一种巧妙的方式：仅复制新的信息。这意味着Elasticsearch知道哪些段已经存储在存储库中，不会

再保存它们。

为了创建新的快照，需要选择一个唯一的名称，并使用下面的命令：

```
curl -XPUT 'localhost:9200/_snapshot/backup/bckp1'
```

上面的命令定义了一个名为bckp1的新快照（给定名称只能有一个快照，Elasticsearch会检查它的唯一性），数据将保存在之前定义的backup库中。该命令立即返回一个响应，如下所示：

```
{"accepted":true}
```

上述响应意味着，快照的进程已经在后台开始并继续。如果你想在实际的快照被创建后才得到响应，可以添加wait_for_completion参数，如下面的例子所示：

```
curl -XPUT 'localhost:9200/_snapshot/backup/bckp2?wait_for_completion=
  true&pretty'
```

上述命令的响应展现了新创建快照的状态：

```
{
  "snapshot" : {
    "snapshot" : "bckp2",
    "indices" : [ "art" ],
    "state" : "SUCCESS",
    "start_time" : "2014-02-22T13:04:40.770Z",
    "start_time_in_millis" : 1393074280770,
    "end_time" : "2014-02-22T13:04:40.781Z",
    "end_time_in_millis" : 1393074280781,
    "duration_in_millis" : 11,
    "failures" : [ ],
    "shards" : {
      "total" : 5,
      "failed" : 0,
      "successful" : 5
    }
  }
}
```

可以看到，Elasticsearch给出了快照过程消耗的时间、它的状态和影响到的索引等信息。

#### 额外参数

快照命令还可以接受以下额外参数。

- indices：想拍下快照的索引名称。
- ignore_unavailable：默认为true。设置为false时，意味着如果indices参数指向了不存在的索引，命令将失败。
- include_global_state：默认值为true，指集群的状态也被写入快照中（除了那些瞬时的设置）。

- partial：快照的成功与否取决于所有分片的可用性。任何一个分片不可用，快照都将失败。设置partial为true时，Elasticsearch值保存可用分片的信息，省略丢失的分片。

使用额外参数的一个示例如下：

```
curl -XPUT 'localhost:9200/_snapshot/backup/bckp?wait_for_
completion=true&pretty' -d '{ "indices": "b*", "include_global_state":
"false" }'
```

### 8.1.3 还原快照

我们已经完成了快照，接下来学习如何从给定的快照中恢复数据。前面说过，可以通过名称指向一个快照。可以使用下面的命令列出所有快照：

```
curl -XGET 'localhost:9200/_snapshot/backup/_all?pretty'
```

先前创建的存储库名称为backup。为了从库中还原一个名为bckp1的快照，执行下面的命令：

```
curl -XPOST 'localhost:9200/_snapshot/backup/bckp1/_restore'
```

执行这个命令的过程中，Elasticsearch取得定义在此快照中的索引，并用快照中的数据创建它们。如果索引已经存在而且未关闭，该命令将失败。这种情况下，你可能会发现只还原某些索引是很方便的，例如：

```
curl -XPOST 'localhost:9200/_snapshot/backup/bck1/_restore?pretty' -d '{
"indices": "c*"}'
```

上述命令只还原以字母c开头的索引，还有以下可用参数。

- ignore_unavailable：与创建快照时相同。
- include_global_state：与创建快照时相同。
- rename_pattern：允许你改变存储在快照中的索引的名称。归功于此，还原的索引将有一个不同的名称。此参数的值是一个正则表达式，定义了源索引的名字，如果模式与索引名字匹配，将发生名称替换。在该模式中，应当使用以rename_replacement参数中所用括号分隔的分组。
- rename_replacement：该参数和rename_pattern一起定义了目标索引名称。使用美元符号和数字可以指向rename_pattern中合适的组。

例如，由于rename_pattern=products_(.*)，只有名称以products_开头的索引将被还原。索引名字的其他部分将使用在替换部分。配合rename_replacement=items_$1，将让products_cars索引被还原成名为items_cars的索引。

### 8.1.4　清理：删除旧的快照

Elasticsearch把快照库的管理交给你。目前，没有自动的清理过程。但不用担心，这很简单。例如，删除之前创建的快照：

```
curl -XDELETE 'localhost:9200/_snapshot/backup/bckp1?pretty'
```

就这么简单。该命令将从backup库中删除名为bckp1的快照。

## 8.2　监控集群的状态和健康度

在应用程序的正常周期中，一个很重要的方面就是监控。这使系统管理员能够检测并预防可能的问题，或至少知道失败时会发生什么。

Elasticsearch提供了非常详细的信息，使你能够检查和监控单个节点或作为一个整体的集群。这包括统计数字、有关服务器的信息、节点、索引和分片。当然，也能够获得整个集群状态的有关信息。在深入提到的API细节之前，请记住，API是复杂的，我们只是描述基本的东西。我们会试着展现什么时候开始，让你在需要非常详细的信息时能知道要找什么。

### 8.2.1　集群健康度API

一个最基本的API是集群健康度API，它允许使用单个HTTP命令得到整个集群的状态信息。例如，执行以下命令：

```
curl 'localhost:9200/_cluster/health?pretty'
```

上述命令返回的响应示例如下所示：

```
{
  "cluster_name" : "es-book",
  "status" : "green",
  "timed_out" : false,
  "number_of_nodes" : 1,
  "number_of_data_nodes" : 1,
  "active_primary_shards" : 4,
  "active_shards" : 4,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 0
}
```

最重要的一个信息是关于集群的状态。在例子中，我们看到集群是green状态。这意味着所有分片已妥善分配，没有错误。

暂停一下，先谈谈作为一个整体的集群什么时候会完全运作。当Elasticsearch能够根据配置

分配所有分片和副本时，集群才全面运作。在这种情况下，集群在green状态。yellow状态意味着我们已经准备好处理请求，因为主分片已经分配，但部分或所有副本还没有。最后一个状态是red，这意味着至少一个主分片没有分配，因此集群还没有准备好。这意味着查询可能返回错误或不完整的结果。

前面的命令也可以用来检查某索引的健康状况。如果想要检查library和map索引的健康度，运行以下命令：

```
curl 'localhost:9200/_cluster/health/library,map/?pretty'
```

#### 1. 控制信息细节

Elasticsearch允许指定一个特殊的level参数，把它指定为cluster（默认）、indices或shards。这样就能够控制由健康度API返回信息的细节。我们已经看过默认的行为。当设置level为indices时，除集群信息外，还将获得每个索引的健康度。参数设置为shards告诉Elasticsearch除了返回我们在示例中看到内容，还要返回每个分片的信息。

#### 2. 额外的参数

除了level参数，还有一些额外参数用来控制健康度API的行为。

第一个参数是timeout。它允许控制命令执行的最长时间。默认值为30s，意味着健康度命令将最长等待30秒钟，然后返回。

wait_for_status参数告诉Elasticsearch返回响应时，集群应该处于什么状态。可以把它设置为green、yellow和red。例如，设置为green时，健康度API调用将返回绿色状态，或者达到timeout时间。

wait_for_nodes参数允许设置返回响应时需要多少节点可用（或者达到timeout时间）。可以设置该参数为整数值，比如3，或者一个简单等式，比如>=3（大于或等于3个节点），<=3（小于或等于3个节点）。

最后一个参数是wait_for_relocating_shard，默认不指定。它告诉Elasticsearch应该重定位多少分片（或者等待timeout时间）。设置该参数为0意味着Elasticsearch应该等待所有重定位分片。

使用以上参数的健康度命令的一个例子如下所示：

```
curl 'localhost:9200/_cluster/health?wait_for_status=green&wait_for_
nodes=>=3&timeout=100s'
```

### 8.2.2 索引统计API

Elasticsearch索引是保存数据的地方，它对大多数部署来说是非常重要的部分。使用_stats

端点上的索引统计API，可以得到关于集群中索引的各种信息。当然，和Elasticsearch大多数API一样，可以发送命令得到所有索引的信息（使用纯_stats端点），或得到某特定索引的信息（例如，library/_stats），或者一次得到几个索引的信息（例如，library,map/_stats）。例如，为了检查本书中使用的map和library索引的统计信息，可以执行如下命令：

```
curl localhost:9200/library,map/_stats?pretty
```

上面的命令将返回超过500行的响应，所以省略它，只描述一下它的结构。除了响应状态和响应时间等信息之外，还可以看到三个对象，分别是primaries、total和indices。indices对象包含library和map索引的信息。primaries对象包含分配在当前节点的主分片信息，total对象包含所有分片（包括副本分片）的所有信息。所有这些对象都包含描述特殊统计的对象，比如docs、store、indexing、get、search、merges、refresh、flush、warmer、filter_cache、id_cache、fielddata、percolate、completion、segments和translog。来讨论存储在这些对象中的信息。

### 1. docs

响应中的docs节点显示索引文档的信息。它看上去可能如下所示：

```
"docs" : {
 "count" : 4,
 "deleted" : 0
}
```

主要信息是count，指文档的数目。从索引中删除文档时，Elasticsearch并不会立即移除这些文档，只把它们标记为删除。这些文档将在段合并过程中被删除。被标记成删除的文档数目将出现在deleted属性中，在合并结束时应该为0。

### 2. store

下一个统计组是store，提供了关于存储的信息。例如，该节点看上去可能如下所示：

```
"store" : {
 "size_in_bytes" : 6003,
 "throttle_time_in_millis" : 0
}
```

主要的信息是关于这个索引（或多个索引）的大小。我们还看到了调节统计值。在系统有I/O性能问题，并且设置了段合并过程中内部操作的限制时，这个信息很有用。

### 3. indexing、get和search

响应中的indexing、get和search节点提供了数据操纵的相关信息：索引删除操作、实时的get和搜索。来看看Elasticsearch返回的下列例子：

```
"indexing" : {
 "index_total" : 11501,
 "index_time_in_millis" : 4574,
 "index_current" : 0,
 "delete_total" : 0,
 "delete_time_in_millis" : 0,
 "delete_current" : 0
},
"get" : {
 "total" : 3,
 "time_in_millis" : 0,
 "exists_total" : 2,
 "exists_time_in_millis" : 0,
 "missing_total" : 1,
 "missing_time_in_millis" : 0,
 "current" : 0
},
"search" : {
 "query_total" : 0,
 "query_time_in_millis" : 0,
 "query_current" : 0,
 "fetch_total" : 0,
 "fetch_time_in_millis" : 0,
 "fetch_current" : 0
}
```

可以看到，所有这些统计具有类似结构。可以看到各种类型的请求花费的总时间（以毫秒为单位）以及请求数，可以用总时间计算单个查询的平均时间。在实时的情况下，get请求中有价值的信息是不成功读取的次数（因为文档缺失）。

#### 4. 额外信息

此外，Elasticsearch提供了下列信息。

- merges：该节点包含Lucene段合并的信息。
- refresh：该节点包含刷新操作的信息。
- flush：该节点包含清理信息。
- warmer：该节点包含预热器的信息，以及它们执行了多久。
- filter_cache：这些是过滤器缓存统计信息。
- id_cache：这些是标识符缓存统计信息。
- fielddata：这些是字段数据缓存统计信息。
- percolate：该节点包含预匹配器使用情况的信息。
- completion：该节点包含自动完成建议器的信息。
- segments：该节点包含Lucene段的信息。
- translog：该节点包含事务日志计数和大小的信息。

### 8.2.3　状态API

另一个得到关于索引信息的方法，是使用_status端点的状态API。返回的信息描述了可用的分片，包含的信息有：哪个分片被认为是主分片，它被分配到了哪个节点，被重分配到了哪个节点（如果是的话），分片的状态（活跃与否），事务日志，合并过程以及刷新和清理统计。

### 8.2.4　节点信息API

节点信息API提供了关于集群中节点的信息，为从该API得到信息，需要发送请求到_nodes端点。

这个API可以使用如下特性获取单个或特定几个节点的信息。

- Node名：如果想得到名为Pulse的节点的信息，可以在_nodes/Pulse REST端点上执行命令。
- Node标识符：如果想得到标识符为ny4hftjNQtuKMyEvpUdQWg的节点的信息，可以在_nodes/ny4hftjNQtuKMyEvpUdQWg端点上执行命令。
- IP地址：如果想得到IP地址为192.168.1.103的节点的信息，可以在_nodes/192.168.1.103 REST端点上执行命令。
- Elasticsearch配置参数：如果想得到node.rack属性等于2的所有节点的信息，可以在_nodes/rack:2 REST端点上执行命令。

该API还允许使用如下方式一次性得到几个节点的信息：

- 模式，例如，_nodes/192.168.1.*或者_nodes/P*；
- 节点枚举，例如，_nodes/Pulse,Slab；
- 模式与节点枚举，例如，/_nodes/P*,S*。

默认情况下，对节点API的请求将返回关于节点的基本信息，比如名称、标识符、地址。通过添加额外参数，可以获得其他信息。可用的参数如下所示。

- settings：此参数用来获取Elasticsearch配置信息。
- os：此参数用来获取服务器的信息，诸如处理器、内存和交换区等。
- process：此参数用来获取进程标识符和可用的文件描述符等信息。
- jvm：此参数用来获取关于Java虚拟机（JVM）的信息，比如内存限制。
- thread_pool：此参数用来获取各种操作的线程池配置。
- network：此参数用来获取网络接口的名称和地址。
- transport：此参数用来获取传输的侦听接口地址。
- http：此参数用来获取HTTP侦听地址。
- plugins：此参数用来获取安装的插件信息。

使用先前描述的API示例如下：

```
curl 'localhost:9200/_nodes/Pulse/os,jvm,plugins?pretty'
```

上述命令除了返回基本信息外，还有操作系统、Java虚拟机和插件的相关信息。当然，所有信息都是针对Pulse节点。

### 8.2.5 节点统计API

节点统计API跟前面描述的节点信息API类似。主要的区别是，节点信息API提供环境信息，而节点统计API告诉我们集群工作时发生过什么。为使用节点统计API，你应该发送命令到/_nodes/stats REST端点。跟节点信息API类似，我们同样可以获取特定节点的信息（比如，_nodes/Pulse/stats）。

默认情况下，Elasticsearch返回所有可用的统计，但可以限制为我们感兴趣的那些。可用的选项如下。

- indices：提供了关于索引的信息，包括大小、文档个数、索引相关的统计、搜索和获取时间、缓存、段合并，等等。
- os：提供了操作系统相关的信息，比如可用磁盘空间、内存、交换分区的使用。
- process：提供了跟Elasticsearch进程相关的包括内存、CPU和文件句柄等的使用情况。
- jvm：提供了关于Java虚拟机内存和垃圾回收统计的信息。
- network：提供了TCP级别的信息。
- transport：提供了传输模块发送和接收数据的信息。
- http：提供了HTTP连接的信息。
- fs：提供了可用磁盘空间和I/O操作统计的信息。
- thread_pool：提供了分配给各种操作线程的状态信息。
- breaker：提供了字段数据缓存断路器的信息。

一个使用该API的示例代码如下所示：

```
curl 'localhost:9200/_nodes/Pulse/stats/os,jvm,breaker?pretty'
```

### 8.2.6 集群状态API

Elasticsearch提供的另一个API是集群状态API。顾名思义，它允许获取关于整个集群的信息（也可以通过在请求中添加local=true，限制只返回本地节点的信息）。用于获取所有信息的基本命令如下所示：

```
curl 'localhost:9200/_cluster/state?pretty'
```

不过，也可以把提供信息限定为特定的度量（用逗号分隔，在REST调用的_cluster/state部分之后指定），以及特定的索引（同样用逗号分隔，在REST调用的_cluster/state/metrics部分之后指定）。下面是一个示例调用，它只返回关于map和library索引的节点相关信息：

```
curl 'localhost:9200/_cluster/state/nodes/map,library?pretty'
```

可使用下面这些度量。

- version：返回关于集群状态版本的信息。
- master_node：返回关于所选主节点的信息。
- nodes：返回节点上的信息。
- routing_table：返回路由相关的信息。
- metadata：返回元数据相关的信息。当指定要获取元数据度量，还可以包含一个额外的参数index_templates=true，将在结果中包含定义的索引模板。
- blocks：返回关于块的信息。

### 8.2.7　挂起任务API

Elasticsearch 1.0中引入的一个API是挂起任务API（pending tasks API），它允许我们坚持哪些任务在等待执行。为获取这个信息，需要发送请求到/_cluster/pending_tasks REST端点。在响应中，我们将看到一组任务，包含任务优先级、队列等待时间等信息。

### 8.2.8　索引段API

我们想提的最后一个API是通过使用/_segments端点的Lucene段API。可以为整个集群或单独的索引执行它。该API提供的信息包括分片、分片的布局，以及Lucene库管理的跟物理索引相连的段。

### 8.2.9　cat API

当然，可以说我们需要的用来诊断和观察集群的所有信息都可以通过提供的API获取。然而，API返回的响应是JSON格式，它很好，但至少对于人类来说，不是特别方便使用。这就是为什么Elasticsearch允许使用一个友好的API：cat API。为使用cat API，需要向_cat REST端点发送一个请求，紧跟着下面的其中一个选项。

- aliases：返回有关别名的信息（8.6节中将介绍别名）。
- allocation：返回分片分配和磁盘使用的信息。
- count：为所有索引或单个索引返回文档个数的信息。
- health：返回集群健康度的信息。

- indices：返回所有索引或单个索引的信息。
- master：返回当选主节点的信息。
- nodes：返回集群拓扑相关信息。
- pending_tasks：返回正在等待执行的任务信息。
- recovery：返回还原过程的视图。
- thread_pool：返回集群范围内的线程池的统计信息。
- shards：返回关于分片的信息。

这可能有点混乱，来看一个返回分片信息的示例命令，如下所示：

```
curl -XGET 'localhost:9200/_cat/shards?v'
```

注意，在请求中包括v参数。这意味着我们希望更详细的信息，例如，包括头部信息。除v参数外，也可以使用help参数，它将返回给定命令的头部描述；还有h参数，它接受一个逗号分隔的列表，指定要包含在响应中的列。

上述命令的响应如下所示：

```
index     shard  prirep  state     docs  store   ip            node
map       0      p       STARTED   4     5.9kb   192.168.1.40  es_node_1
library   0      p       STARTED   9     11.8kb  192.168.56.1  es_node_2
```

可以看到，我们有两个索引，每个都有一个分片。还看到分片的ID，也就是说，它是不是主分片，以及它的状态、文档数、大小、节点的IP地址、节点名称等。

### 限制返回信息

一些cat API命令允许限制它们返回的信息。例如，别名调用允许我们通过添加别名来得到特定别名的信息，就像下面的命令：

```
curl -XGET 'localhost:9200/_cat/aliases/current_index'
```

我们总结一下允许限制信息的命令。

- aliases：通过在请求中添加别名，来限定获取特定别名的信息。
- count：通过在请求中添加我们感兴趣的索引名字，来限定获取特定索引的信息。
- indices：同count一样，限定只获取特定索引的信息。
- shards：通过在请求中添加我们感兴趣的索引名字，来限定获取特定索引的信息。

## 8.3　控制集群的再平衡

默认情况下，Elasticsearch试图把分片和其副本在集群中均衡分布。在大多数情况下这种行为是好的，但有时我们想控制此行为。本节将深入介绍如何避免集群再平衡（rebalancing），以及如何控制这一过程的行为。

假设你的网络可以处理很高的流量，或者相反，网络被大量使用，你希望可以避免太多压力。另一个例子是，你可能想在整个集群重启后，减少I/O子系统的压力，并且同时你要减少分片和副本的初始化。这只是两个再平衡控制可能很方便的例子。

### 8.3.1　再平衡

再平衡是在集群的不同节点之间移动分片的过程。我们已经提到，在大多数情况下它是好的，但有时你可能想完全避免这种情况。如果我们定义且希望保持分片放置，就要避免再平衡。然而，默认情况下，当集群状态变化，或者Elasticsearch认为需要再平衡时，它会尽量对集群进行再平衡。

### 8.3.2　集群的就绪

我们已经知道，索引可以由分片和副本构成。主分片（或者简称分片）用于新文档被编入索引以及更新或删除，或者只是索引发生任何变化时。我们的副本从主分片获取数据。

你可以认为，当所有主分片都被分配在集群中的节点上，也就是一旦达到黄色的健康状态时，集群就已经就绪了。然而，此时Elasticsearch还可能在初始化其他分片：副本。但是，你已经可以使用集群，并确保可以搜索整个数据集，也可以发送索引更改命令。之后，这些都将被正确处理。

### 8.3.3　集群再平衡设置

Elasticsearch允许控制再平衡过程，通过设置elasticsearch.yml文件中的几个属性，或使用Elasticsearch REST API（在8.8节中描述）。

#### 1. 控制再平衡何时开始

可以通过设置`cluster.routing.allocation.allow_rebalance`属性来指定再平衡何时开始。该属性可以设置为如下几个值。

- `always`：该值表明再平衡可以在需要时随时开始。
- `indices_primaries_active`：该值表明当所有的主分片都初始化后，再平衡才会开始。

- indices_all_active：该值是默认设置，意味着所有分片和副本都初始化后，再平衡才会开始。

#### 2. 控制同时在节点中移动的分片数量

可以设置cluster.routing.allocation.cluster_concurrent_rebalance属性来指定整个集群中同时可以在节点间移动的分片数量。如果你的集群由很多节点组成，可以提高这个值，默认值为2。

#### 3. 控制单个节点上同时初始化的分片数量

cluster.routing.allocation.node_concurrent_recoveries属性可以用来设置Elasticsearch在单个节点上一次可以初始化多少分片。请注意分片的还原过程是非常耗I/O的，所以你可能要避免太多的分片同时被还原。该属性值默认跟上一个属性一样，为2。

#### 4. 控制单个节点上同时初始化的主分片数

可以设置cluster.routing.allocation.node_initial_primaries_recoveries属性来控制单个节点上一次可以初始化多少主分片。

#### 5. 控制分配的分片类型

使用cluster.routing.allocation.enable属性，可以控制允许分配哪种类型的分片。该属性可以使用如下值。

- all：这是默认值，告诉Elasticsearch所有类型的分片都可以被分配。
- primaries：告诉Elasticsearch它应该只分配主分片，而不要分配副本。
- new_primaries：告诉Elasticsearch只分配新创建的主分片。
- none：这完全禁用了分片的分配。

#### 6. 控制单个节点上的并发流数目

indices.recovery.concurrent_streams属性允许控制在一个节点上一次可以打开多少流，以便从目标分片中恢复一个分片。它的默认值为3。如果你的网络和节点可以处理更多，就可以提高这个值。

## 8.4 控制分片和副本的分配

Elasticsearch集群内部的索引，可以由许多分片建成，每个分片可以有多个副本。因为单一的索引可以有多个分片，可以处理索引大到无法存入到单台机器的情况。可能有其他原因，比如与存储相关的内存和CPU等。因为每个分片可以有多个副本，可以通过把副本散布到多台服务器

上，来处理更高的查询。可以说通过使用分片和副本，可以横向扩展Elasticsearch。然而，Elasticsearch必须找出在集群的什么地方放置分片和副本，即每个分片或副本应放在哪些服务器或节点上。

## 8.4.1　显式控制分配

假设希望把索引放置在不同的集群节点上。例如，把一个叫shop的索引放置在某些节点上，第二个叫users的索引放置在其他节点上。把最后一个名为promotions的索引放置在users和shop索引放置的所有节点上。需要这样做可能是由于性能原因。我们知道安装过Elasticsearch的一些服务器比其他服务器更强大。使用默认Elasticsearch行为，我们不知道分片和副本将放置在哪，但幸好，Elasticsearch允许我们控制它。

### 1. 指定节点参数

所以让我们把集群分为两个区域。我们说“区”，但它可以是任何你喜欢的名称，我们只是喜欢用“区”。假设有四个节点，希望把更强大的编号为1和2的节点放置在一个叫zone_one的区；编号3和4的节点资源较少，放在叫zone_two的区域。

### 2. 配置

为了实现我们描述的索引分布，在节点1和节点2（较强的节点）的elasticsearch.yml配置文件中添加node.zone: zone_one属性。在节点3和节点4（较弱的节点）的elasticsearch.yml配置文件中添加类似的属性：node.zone: zone_two。

### 3. 索引的创建

现在创建索引。首先创建shop索引。将此索引放置到更强的节点。为此可以运行以下命令：

```
curl -XPUT 'http://localhost:9200/shop' -d '{
 "settings" : {
  "index" : {
   "routing.allocation.include.zone" : "zone_one"
  }
 }
}'
```

上述命令将创建shop索引并指定index.routing.allocation.include.zone属性的值为zone_one，这意味着我们希望把shop索引放到node.zone属性等于zone_one的节点上。

为users索引执行类似的步骤：

```
curl -XPUT 'http://localhost:9200/users' -d '{
 "settings" : {
  "index" : {
   "routing.allocation.include.zone" : "zone_two"
```

```
  }
 }
}'
```

不过这一次，我们指定希望把users索引放到node.zone属性等于zone_two的节点上。

最后，promotions索引应该放到上述所有节点，因此使用下列命令来创建和配置这个索引：

```
curl -XPOST 'http://localhost:9200/promotions'
curl -XPUT 'http://localhost:9200/promotions/_settings' -d '{
 "index.routing.allocation.include.zone" : "zone_one,zone_two"
}'
```

这次使用了不同的命令集。第一个命令创建索引，第二个命令更新index.routing.allocation.include.zone属性的值。这样做是为了说明可以用这种方式来做。

#### 4. 排除节点的分配

既然可以指定索引应该放在哪个节点，就可以指定应该排除哪些节点。参考之前的例子，如果希望名为pictures的索引不放在node.zone属性等于zone_one的节点上，可以执行以下命令：

```
curl -XPUT 'localhost:9200/pictures/_settings' -d '{
 "index.routing.allocation.exclude.zone" : "zone_one"
}'
```

注意，这次使用index.routing.allocation.exclude.zone属性，而不是index.routing.allocation.include.zone属性。

#### 5. 节点需求属性

除了节点的包含和排除规则，还可以指定分片必须匹配某种规则才能分配到给定节点上。不同的是，使用index.routing.allocation.include属性时，索引将被放置到与该属性中至少一个值匹配的节点上。而使用index.routing.allocation.require属性值时，Elasticsearch将把索引放置到与该属性的所有值都匹配的节点上。例如，我们已经为pictures索引做了如下设置：

```
url -XPUT 'localhost:9200/pictures/_settings' -d '{
 "index.routing.allocation.require.size" : "big_node",
 "index.routing.allocation.require.zone" : "zone_one"
}'
```

执行以上命令后，Elasticsearch将只会把pictures索引的分片分配到node.size属性等于big_node且node.zone属性等于zone_one的节点上。

#### 6. 使用IP地址分配分片

除了在节点的配置中添加一个特殊的参数，也可以使用IP地址来指定应该包含或排除哪些节点用来做分片和副本的分配。为此，替换index.routing.allocation.include.zone或index.routing.allocation.exclude.zone属性的zone部分，而改用_ip。如果希望把shop

索引只放置到IP地址为10.1.2.10和10.1.2.11的节点上，执行以下命令：

```
curl -XPUT 'localhost:9200/shop/_settings' -d '{
 "index.routing.allocation.include._ip" : "10.1.2.10,10.1.2.11"
}'
```

7. 基于磁盘的分片分配

除了上面描述的分片过滤方法外，Elasticsearch 1.0引入了一个额外的基于磁盘的方法，它允许基于节点的磁盘使用情况来设置分配规则，因此不会有耗尽磁盘空间或类似的问题。

(1) 启用基于磁盘的分片分配

基于磁盘的分片分配默认是禁用的。可以通过设置cluster.routing.allocation.disk.threshold_enabled属性为true来启用它。可以在elasticsearch.yml文件中设置，或者使用集群设置API动态设置（8.8节会介绍）：

```
curl -XPUT localhost:9200/_cluster/settings -d '{
 "transient" : {
  "cluster.routing.allocation.disk.threshold_enabled" : true
 }
}'
```

(2) 配置基于磁盘的分片分配

以下三个属性可以用来控制基于磁盘的分片分配的行为，它们都可以动态更新，或者在elasticsearch.yml配置文件中设置。

第一个属性是cluster.info.update.interval，默认值为30秒，定义了Elasticsearch更新节点上磁盘使用信息的时间间隔。

第二个属性是cluster.routing.allocation.disk.watermark.low，默认值为0.70。这意味着Elasticsearch不会在磁盘空间被使用超过70%的节点上分配新的分片。

第三个属性是cluster.routing.allocation.disk.watermark.high，默认值为0.85。意味着Elasticsearch对磁盘使用大于等于85%的节点将开始重新分配分片。

cluster.routing.allocation.disk.watermark.low 和 cluster.routing.allocation.disk.watermark.high属性都可以设置成一个百分比（比如0.60，表示60%），或者一个绝对值（600 mb，表示600兆字节）

## 8.4.2　集群范围的分配

除了在索引级别上指定分配的包含和排除规则（到目前为止我们讨论的），还可以在集群中的所有索引上指定。如果希望把所有新索引都放置到IP地址为10.1.2.10和10.1.2.11的节点上，执

行如下命令：

```
curl -XPUT 'localhost:9200/_cluster/settings' -d '{
 "transient" : {
  "cluster.routing.allocation.include._ip" : "10.1.2.10,10.1.2.11"
 }
}'
```

我们注意到，命令发送到_cluster/settings REST端点，而不是INDEX_NAME/_settings端点。当然，可以使用在索引级别上的所有包含、排除、需求规则。

请注意，集群的瞬时和永久属性在8.8节讨论。

### 8.4.3 每个节点上的分片和副本数量

除了指定分片和副本的分配外，还可以指定单一节点上为单一索引最多可以放置多少分片。如果希望shop索引在每个节点上只有一个分片，执行以下命令：

```
curl -XPUT 'localhost:9200/shop/_settings' -d '{
 "index.routing.allocation.total_shards_per_node" : 1
}'
```

该属性可以放在elasticsearch.yml文件中，或者使用上述命令更新生产环境的索引。请记住，如果Elasticsearch无法分配所有的主分片，你的集群将维持在红色状态。

### 8.4.4 手动移动分片和副本

我们想讨论的最后一件事是手动在节点间移动分片的能力。这可能有用，例如，在关掉一个节点之前，你想把该节点上的所有分片移动到其他地方。Elasticsearch公开了_cluster/reroute REST端点，允许我们控制这个。

有以下可用的操作：

- 把分片从一个节点移到另一个节点；
- 取消分片的分配；
- 强制分片的分配。

现在让我们仔细看看上述操作。

#### 1. 移动分片

假设有两个节点，分别为es_node_one和es_node_two。除此之外，我们的shop索引有两个分片被Elasticsearch放置在第一个节点上。现在，我们想把第二个分片移动到第二个节点上。为此，可以执行以下命令：

```
curl -XPOST 'localhost:9200/_cluster/reroute' -d '{
 "commands" : [ {
  "move" : {
   "index" : "shop",
   "shard" : 1,
   "from_node" : "es_node_one",
   "to_node" : "es_node_two"
  }
 } ]
}'
```

我们指定了move命令，它允许移动由index属性指定的索引的分片（和副本）。shard属性是要移动的分片的编号。最后，from_node属性指定了希望从哪个节点上移动分片，to_node属性指定了希望把分片放置到哪个节点上。

### 2. 取消分片的分配

如果想取消正在进行的分配过程，可以执行cancel命令，并指定想取消分片的索引、节点和分片。例如以下命令：

```
curl -XPOST 'localhost:9200/_cluster/reroute' -d '{
  "commands" : [ {
    "cancel" : {
    "index" : "shop",
    "shard" : 0,
    "node" : "es_node_one"
    }
  } ]
}'
```

上述命令将取消es_node_one节点上shop索引编号为0的分片分配。

### 3. 强制分片的分配

除了取消和移动分片和副本，还可以分配一个未分配的分片到指定节点上。如果我们的users索引有个编号为0的未分配分片，想把它分配到es_node_two节点上，执行以下命令：

```
curl -XPOST 'localhost:9200/_cluster/reroute' -d '{
  "commands" : [ {
    "allocate" : {
    "index" : "users",
    "shard" : 0,
    "node" : "es_node_two"
   }
  } ]
}'
```

### 4. 每个HTTP请求多个命令

当然，也可以在单个HTTP请求中包含多个命令，例如，考虑以下命令：

```
curl -XPOST 'localhost:9200/_cluster/reroute' -d '{
 "commands" : [
  {"move" : {"index" : "shop", "shard" : 1, "from_node" : "es_node_one",
"to_node" : "es_node_two"}},
  {"cancel" : {"index" : "shop", "shard" : 0, "node" : "es_node_one"}}
 ]
}'
```

## 8.5 预热

有时，可能需要为了处理查询而准备Elasticsearch。也可能因为你严重依赖字段数据缓存，需要在生产查询到达之前加载它们，或者你可能想预热操作系统I/O缓存。不管什么原因，Elasticsearch允许为类型和索引定义预热查询（warning query）。

### 8.5.1 定义一个新的预热查询

预热查询跟普通查询没什么区别，只是它存储在Elasticsearch一个特殊的名为_warmer的索引中。假设想使用下面的查询来做预热：

```
{
  "query" : {
    "match_all" : {}
  },
  "facets" : {
    "warming_facet" : {
      "terms" : {
        "field" : "tags"
      }
    }
  }
}
```

为了把上述查询存储为library索引的预热查询，执行以下命令：

```
curl -XPUT 'localhost:9200/library/_warmer/tags_warming_query' -d '{
  "query" : {
    "match_all" : {}
  },
  "facets" : {
    "warming_facet" : {
      "terms" : {
        "field" : "tags"
      }
    }
  }
}'
```

上诉命令将我们的查询注册为一个名为tags_warming_query的预热查询。你的索引可以

有多个预热查询，但每一个查询都需要一个唯一的名称。

我们不仅可以对整个索引定义预热查询，还可以为索引中的特定类型定义。为了把前面展现的查询存为library索引中book类型的预热查询，执行之前的命令，但不再是/library/_warmer URI，而是/library/book/_warmer。所以，整个命令将如下所示：

```
curl -XPUT 'localhost:9200/library/book/_warmer/tags_warming_query' -d '{
  "query" : {
    "match_all" : {}
  },
  "facets" : {
    "warming_facet" : {
      "terms" : {
        "field" : "tags"
      }
    }
  }
}'
```

添加一个预热查询后，Elasticsearch允许一个新段执行搜索之前，会在那个段上执行定义的预热查询。它允许Elasticsearch和操作系统缓存数据，以此来加速搜索。

正如1.1节中介绍的，Lucene把索引分为多个段，段一旦写入不再修改。每个新的提交操作都会创建一个新段（如果段的数量太大，将会被合并），Lucene使用它来搜索。

## 8.5.2　获取定义的预热查询

为了获取索引中的特定预热查询，我们只需知道它的名称。如果想获取library索引中名为tags_warming_query的预热查询，执行下面的命令：

```
curl -XGET 'localhost:9200/library/_warmer/tags_warming_query?pretty=true'
```

Elasticsearch返回的结果将如下所示（注意我们使用了pretty=true参数让响应更易阅读）：

```
{
  "library" : {
    "warmers" : {
      "tags_warming_query" : {
        "types" : [ ],
        "source" : {
          "query" : {
            "match_all" : { }
          },
          "facets" : {
            "warming_facet" : {
              "terms" : {
                "field" : "tags"
```

```
                }
              }
            }
          }
        }
      }
    }
}
```

也可以使用下面的命令来获取索引和类型的所有预热查询：

```
curl -XGET 'localhost:9200/library/_warmer'
```

最后，还可以获取所有以特定前缀开头的预热查询。例如，想得到library索引上所有以tags前缀开头的预热查询，执行下面的命令：

```
curl -XGET 'localhost:9200/library/_warmer/tags*'
```

### 8.5.3 删除一个预热查询

删除预热查询跟获取预热查询非常类似，只需使用DELETE这个HTTP方法。为了从索引中删除一个特定的预热查询，我们只需知道它的名称。例如，从library索引中删除名为tags_warming_query的预热查询，执行以下命令：

```
curl -XDELETE 'localhost:9200/library/_warmer/tags_warming_query'
```

也可以使用下面命令来删除索引上的所有预热查询：

```
curl -XDELETE 'localhost:9200/library/_warmer/_all'
```

最后，也可以删除所有以给定前缀开头的预热查询。例如，要删除library索引上的所有以tags前缀开头的预热查询，可执行以下命令：

```
curl -XDELETE 'localhost:9200/library/_warmer/tags*'
```

### 8.5.4 禁用预热功能

为了完全禁用预热查询，但仍把它们保存在_warmer索引中，你应该设置index.warmer.enabled属性为false（设置为true则启用预热功能）。该属性可以在elasticsearch.yml文件中设置，也可以在生产集群上使用REST API。

如果想禁用library索引上的预热功能，执行以下命令：

```
curl -XPUT 'http://localhost:9200/library/_settings' -d '{
  "index.warmer.enabled" : false
}'
```

### 8.5.5　查询的选择

你可能会问哪个查询应该用做预热查询。通常要选择执行起来昂贵的和需要填充缓存的查询。因此，可能会选择基于索引中的字段做切面和排序的查询。除此以外，父子查询和包括常用过滤器的查询也可考虑。你也可以通过查看日志选择其他查询，找到性能表现不尽人意的，这些也可以是很好的预热候选对象。

假设在elasticsearch.yml文件中设置了以下的日志配置：

```
index.search.slowlog.threshold.query.warn: 10s
index.search.slowlog.threshold.query.info: 5s
index.search.slowlog.threshold.query.debug: 2s
index.search.slowlog.threshold.query.trace: 1s
```

并且在logging.yml配置文件中设置了以下日志级别：

```
logger:
  index.search.slowlog: TRACE, index_search_slow_log_file
```

注意，`index.search.slowlog.threshold.query.trace`属性设置成1s，而日志级别`index.search.slowlog`属性设置成`TRACE`。这意味着每当一个查询执行时间查过1秒（在分片上，不是总时间）时，它将被记录到慢查询日志文件中（日志文件由logging.yml配置文件中的`index_search_slow_log_file`节点指定）。例如，慢查询日志文件中可能找到下面的条目：

```
[2013-01-24 13:33:05,518][TRACE][index.search.slowlog.query] [Local
test] [library][1] took[1400.7ms], took_millis[1400], search_
type[QUERY_THEN_FETCH], total_shards[32], source[{"query":{"match_
all":{}}}], extra_source[]
```

可以看到，前面的日志行包含查询时间、搜索类型和搜索源本身，它显示了执行的查询。

当然，你的配置文件中可能有不同的值，但慢查询日志可以是一个很有价值的源，从中可以找到执行时间过长的、可能需要定义预热的查询，它们可能是父子查询，需要获取一些标识符来提高性能，或者你第一次使用了一些费时的过滤器?

应该记住，不要让你的Elasticsearch集群加载过多的预热查询，因为最终你可能会花太多的时间预热，而不是处理你的生产查询。

## 8.6　使用索引别名来简化你的日常工作

当在Elasticsearch中使用多个索引时，你可能有时会忘记它们。想象你在索引中存储日志的场景。通常，日志消息的数量相当大，因此，把数据划分是一个好的解决方案。一种逻辑划分方法是为每天的日志创建一个索引（如果你对管理日志的开源方案感兴趣，可以看看

http://logstash.net上的Logstash）。但一段时间后，如果保存了所有索引，就会开始头疼如何管理它们。应用程序需要管理所有这些信息，比如发送数据到哪个索引，查询哪个索引，等等。通过使用别名，可以改变这个状况，使用一个名字来跟多个索引打交道，就像使用一个索引一样。

### 8.6.1 别名

什么是索引别名？它是一个或多个索引的一个附加名称，允许使用这个名称来查询索引。一个别名可以对应多个索引，反之亦然，一个索引可以是多个别名的一部分。然而，请记住，你不能使用对应多个索引的别名来进行索引或实时的GET操作。如果你这样做，Elasticsearch将抛出一个异常（但可以使用对应单个索引的别名来进行索引操作），因为Elasticsearch不知道应该把索引建立到哪个索引上，或从哪个索引获取文档。

### 8.6.2 创建别名

为创建一个索引别名，需要在_aliases REST端点上执行一个HTTP POST方法。例如，如下的请求将创建一个名为week12的别名，它包含day10、day11和day12这些索引。

```
curl -XPOST 'localhost:9200/_aliases' -d '{
  "actions" : [
    { "add" : { "index" : "day10", "alias" : "week12" } },
    { "add" : { "index" : "day11", "alias" : "week12" } },
    { "add" : { "index" : "day12", "alias" : "week12" } }
  ]
}'
```

如果我们的Elasticsearch集群中不存在week12别名，上述命令将创建它。如果存在，该命令将只是把指定的索引添加进去。

之前，执行一个跨三个索引的搜索是这样的：

```
curl -XGET 'localhost:9200/day10,day11,day12/_search?q=test'
```

如果一切顺利，现在可以这样执行：

```
curl -XGET 'localhost:9200/week12/_search?q=test'
```

这不是更好吗？

### 8.6.3 修改别名

当然，你可以从别名中移除索引。这与添加索引到别名类似，但使用remove命令，而不是add。例如，为了从week12中移除day9索引，执行以下命令：

```
curl -XPOST 'localhost:9200/_aliases' -d '{
  "actions" : [
    { "remove" : { "index" : "day9", "alias" : "week12" } }
  ]
}'
```

### 8.6.4　合并命令

add和remove命令可以在一个请求中发送。例如，想合并之前发送的所有命令到一个单一的请求中，可以发送以下命令：

```
curl -XPOST 'localhost:9200/_aliases' -d '{
  "actions" : [
    { "add" : { "index" : "day10", "alias" : "week12" } },
    { "add" : { "index" : "day11", "alias" : "week12" } },
    { "add" : { "index" : "day12", "alias" : "week12" } },
    { "remove" : { "index" : "day9", "alias" : "week12" } }
  ]
}'
```

### 8.6.5　获取所有别名

除了对别名添加或移除索引，使用Elasticsearch的应用程序，可能需要获取集群中的所有别名或跟一个索引连接的所有别名。为了获取这些别名，发送一个HTTP GET命令。例如，下面的第一个命令获取day10索引的所有别名，第二个命令获取所有可能的别名：

```
curl -XGET 'localhost:9200/day10/_aliases'
curl -XGET 'localhost:9200/_aliases'
```

第二个命令返回的响应如下：

```
{
  "day10" : {
    "aliases" : {
      "week12" : { }
    }
  },
  "day11" : {
    "aliases" : {
      "week12" : { }
    }
  },
  "day12" : {
    "aliases" : {
      "week12" : { }
    }
  }
}
```

### 8.6.6 移除别名

可以使用_alias端点移除一个别名。例如，发送下面命令将从data索引移除client别名：

```
curl -XDELETE localhost:9200/data/_alias/client
```

### 8.6.7 别名中的过滤

可以像在SQL数据库中使用视图一样来使用别名。你可以使用完整的查询DSL（第2章详细讨论过），将你的查询应用到所有的count、search、delete，等等。

来看一个例子。假设想要一个别名返回特定客户的数据，以便在应用程序中使用。客户标识符存在clientId字段上，我们对客户12345感兴趣。所以，在数据索引中创建一个名为client的别名，它自动在clientId上应用一个过滤器：

```
curl -XPOST 'localhost:9200/_aliases' -d '{
  "actions" : [
  {
    "add" : {
      "index" : "data",
      "alias" : "client",
      "filter" : { "term" : { "clientId" : "12345" } }
    }
  } ]
}'
```

所以，当使用该别名时，你的请求将总是被一个词条查询过滤，确保所有文档的clientId字段都等于12345。

### 8.6.8 别名和路由

跟在别名中使用过滤器类似，可以在别名中添加路由值。假设我们在使用基于用户标识符的路由，且想在别名中使用相同的路由值，则在名为client的别名中，我们将在查询中使用路由值12345、12346、12347，而在索引建立中只使用12345。为此，使用下面的命令创建别名：

```
curl -XPOST 'localhost:9200/_aliases' -d '{
  "actions" : [
  {
    "add" : {
      "index" : "data",
      "alias" : "client",
      "search_routing" : "12345,12346,12347",
      "index_routing" : "12345"
    }
  } ]
}'
```

这样，使用client别名索引数据时，将使用index_routing属性的值。在查询时，将使用search_routing属性值。

还有一件事。请看发送到上述别名的如下查询：

```
curl -XGET 'localhost:9200/client/_search?q=test&routing=99999,12345'
```

它将使用12345作为路由值。这是因为Elasticsearch将使用search_routing属性值和查询路由参数值的共同值，在我们的例子中是12345。

## 8.7　Elasticsearch 插件

在本书不同的地方，我们使用了不同的Elasticsearch插件。你可能还记得6.5节描述过的，脚本中使用的额外编程语言和对附件的支持。本节将介绍插件如何工作以及安装。

### 8.7.1　基础知识

Elasticsearch插件位于plugins目录的各自子目录中。如果从网站上下载一个新的插件，你可以用该插件名创建一个新目录，并把插件存档解压到这个目录中。还有个更方便的安装方法：使用插件脚本。我们在本书中已经用过几次，所以现在该描述这个工具了。

Elasticsearch有两种主要类型的插件。这两种类型可以基于其内容来分类：Java插件和站点插件（site plugins）。Elasticsearch把站点插件当成被内置的HTTP服务器处理的文件集，处于/_plugin/plugin_name/URL（比如/_plugin/bigdesk）下面。此外，任何一个没有Java内容的插件将被自动视为站点插件，就这么简单。在Elasticsearch看来，站点插件不会改变Elasticsearch的行为。Java插件通常包含.jar文件，被es-plugin.properties文件扫描。该文件包含主要类的信息，Elasticsearch使用该类作为入口来配置插件并扩展Elasticsearch的功能。Java插件可以包含站点部分，被内置的HTTP服务器使用（跟站点插件一样），这部分需要放置在_site目录中。

### 8.7.2　安装插件

默认情况下，插件从网站下载。如果该网站找不到插件，将检查Maven Central（http://search.maven.org/）、Maven Sonatype(https://repository.sonatype.org/)和GitHub（https://github. com/)等库。插件工具假定给定的插件地址由组织名称、插件名称和版本号组成。看看下面的命令：

```
bin/plugin -install elasticsearch/elasticsearch-lang-javascript/2.0.0.RC1
```

上述命令将安装一个插件，该插件允许使用额外的脚本语言：JavaScript。选择该插件的2.0.0.RC1版本。也可以省略版本号，那样，Elasticsearch将尝试寻找跟Elasticsearch版本号一样的版本或者最新的主版本。

我们期待命令的结果，下面是执行上述命令的一个示例结果：

```
-> Installing elasticsearch/elasticsearch-lang-javascript/2.0.0.RC1...
Trying http://download.elasticsearch.org/elasticsearch/elasticsearch-
lang-javascript/elasticsearch-lang-javascript-2.0.0.RC1.zip...
Downloading ..........................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................DONE
Installed elasticsearch/elasticsearch-lang-javascript/2.0.0.RC1 into /
opt/elasticsearch-1.0.0/plugins/lang-javascript
```

如果你编写自己的插件，没有访问上述网站的权限，也没有问题。插件工具提供了`-url`选项，允许为插件设置包括本地文件系统（使用`file://`前缀）在内的任何地址。例如，下面的命令将安装存档在本地文件系统的`/tmp/elasticsearch-lang-javascript-2.0.0.RC1.zip`里的插件：

```
bin/plugin -install lang-javascript -url file:///tmp/elasticsearch-lang-
javascript-2.0.0.RC1.zip
```

### 8.7.3 移除插件

移除插件跟移除目录一样简单。你也可以使用插件工具来做。例如，为了移除`river-mongodb`插件，可以执行下面的命令：

```
bin/plugin -remove river-mongodb
```

你需要重启Elasticsearch节点，以让插件的安装或移除生效。

## 8.8 更新设置 API

Elasticsearch允许在elasticsearch.yml文件中指定各种参数来调优。但你应该把这个文件当做默认设置，可以在运行时通过Elasticsearch REST API修改。

为了设置其中一个属性，需要使用HTTP PUT方法，发送一个合适的请求到_cluster/setting URI。我们有两个选择：瞬时和永久的属性设置。

第一个，瞬时，将只设置属性直到第一次重启。为此，发送下面的命令：

```
curl -XPUT 'localhost:9200/_cluster/settings' -d '{
  "transient" : {
    "PROPERTY_NAME" : "PROPERTY_VALUE"
  }
}'
```

可以看到，在上面的命令中，我们使用了名为transient的对象，并在其中添加属性定义。这意味着该属性值将生效，直到重新启动。如果希望属性设置在重启之后永久生效，使用名称persistent，而不是transient。

任何时候，都可以使用下列命令来获取这些设置：

```
curl -XGET localhost:9200/_cluster/settings
```

## 8.9　小结

本章介绍了如何为集群创建备份；创建一个备份存储库、创建了备份，并管理它们。我们了解到如何使用Elasticsearch API监控集群，什么是cat API以及为什么它更方便使用。我们还控制了分片的分配，学会了如何在集群中移动分片和控制集群再平衡。我们使用预热功能为生产查询准备集群，学到别名如何帮助我们管理集群中的数据。最后，本章研究了Elasticsearch插件，以及如何使用Elasticsearch提供的更新设置API。

本书内容到此结束了。我们希望你有一个很好的阅读体验并觉得这是本有趣的书。我们真的希望你能有所收获，并从此发现在日常工作中使用Elasticsearch更容易了。作为本书的作者和Elasticsearch用户，我们试图带给读者最好的阅读体验。当然，Elasticsearch还有很多书中没有描述的内容，特别是涉及监控、管理功能和API的内容。图书的页面毕竟是有限的，如果把一切都描述得很详细，这本书就得一千多页了。此外，我们恐怕也无法把一切都描述得足够详细。Elasticsearch不仅是用户友好的，还提供了大量配置选项、查询的可能性等，因此，我们必须选择哪些功能要详细说明，哪些只是一带而过，哪些要完全跳过。希望本书对主题的选择是对的。

还要说的是，记住Elasticsearch在不断演化。在写这本书期间，我们经历了一些稳定的版本，最终发布了1.0.0和1.0.1。即便在这之前，我们也知道会有新功能和改进。如果你想跟进正在添加的新功能，请一定定期检查上新版本的发布说明。我们也会把自认为值得一提的新功能写在上，所以如果你感兴趣，请时常访问此网站。