链滴

22. Testing

作者：felayman
原文链接：https://ld246.com/article/1512829871934
来源网站：链滴

测试(Testing)

- 原文地址: Testing
- 版本： 6.0.0

Testing

- Java Testing Framework（测试框架）

Java Testing Framework（测试框架）

测试是您的应用程序的重要组成部分，由于 information retrieval（信息检索）本身已经是一个复杂话题了，因此我们在设置使用 elasticsearch 的测试基础架构的时候不应该有额外的复杂性。

这是我们决定在发行版中发布一个附加文件的主要原因，它允许您使用与 elasticsearch core 相同的试基础架构。

testing framework（测试框架）允许您设置具有多个节点的集群，以便检查您的代码是否涵盖在集中运行所需的所有的内容。该框架阻止您自己编写复杂代码来启动，停止或者管理集群中的多个节点。

另外，还有一个非常重要的功能叫做randomized testing（随机测试），您可以免费获取它，因为它 elasticsearch 基础架构的一部分。

(why randomized testing) 为什么随机测试

why randomized testing?

随机测试的关键概念不是为每个测试用例使用相同的输入值，但是在出现故障的情况下仍然可以重现们。这允许测试使用大不相同的输入变量，以确保您的实现实际上独立于您提供的测试数据。

所有测试都使用随机测试项目 (randomized-testing project) 提供的一个自定义的 junit runner — RandomizedRunner 运行。如果您对正在使用的实现感兴趣，请查看 RandomizedTesting webpa e 。

Using the elasticsearch test classes (使用 elasticsearch 测试类)

使用 elasticsearch 测试类

首先，您需要在项目中包含测试依赖项 (testing dependency)，以及已添加的 elasticsearch depe dency (elasticsearch 依赖项)。如果您使用 maven 和 它的 pom.xml，它看起来像下面这样：

```
<dependencies>
  <dependency>
    <groupId>org.apache.lucene</groupId>
    <artifactId>lucene-test-framework</artifactId>
    <version>${lucene.version}</version>
    <scope>test</scope>
  </dependency>
```

```
  <dependency>
    <groupId>org.elasticsearch.test</groupId>
    <artifactId>framework</artifactId>
    <version>${elasticsearch.version}</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

用相应的 elasticsearch 版本及其附带的 lucene 版本替换 elasticsearch 版本和 lucene 版本。

我们提供了几个可以在您自己的测试类中继承的类，它们提供：

- pre-defined loggers (预定义的日志输出器)
- randomized testing infrastructure (随机测试的基础工具)
- a number of helper methods (一系列帮助方法)

unit tests（单元测试）

单元测试

如果您将进行的是一次完全隔离的单元测试，可以使用 ESTestCase，而无需运行 elasticsearch clust r （弹信收索集群）。如果您要测试lucene功能，请使用 ESTestCase，如果您要测试具体的 token st eams ，请使用 ESTokenStreamTestCase 这个类。这些特定的类会执行附加的检查，确保在测试运后不会发生

资源泄露。

integreation test（集成测试）

integration test （集成测试）

只有所有的运算集群都运行后，集成测试才能开始运行。和单元测试相比，集成测试花费的时间更多但是集成测试基础结构通过仅重新开启整个集群试图将花费的时间最小化（如果配置已经确定了）。

如果您要进行集成测试，则您的测试类需要继承 ESIntegTestCase类。通过继承此类，您只需要确保少有一定数量的节点启动，而不再需要在测试中手动启动弹性搜索节点。您通过在测试运行中指定不的系统属性，完成大量的配置集成测试。 有关更多信息，请参阅源存储库中的 TESTING.asciidoc 。

number of shards （分片数）

集成测试期间，除非通过在创建索引时重写索引设置，否则索引创建的分片数量在1~10 之间随机生。 除非一定需要，否则默认规则不会指定碎片数量，以便未来每次测试都将一直使用不同的分片。者，您可以重写 numberOfShards() 方法。 这方法同样适用于 numberOfReplicas()。

generic helper methods （通用帮助方法）

ESIntegTestCase 中有几种帮助方法，这将使您的测试更简洁。

方法名称	描述

refresh()	刷新集群中的所有索引
ensureGreen()	确定集群状态处于绿色健康状态
等待搬迁。 默认等待30秒，30秒后，超时失败。	
ensureYellow()	确定集群状态处于黄色健康状态
同样等待30秒，30秒后，超时失败。	
createIndex(name)	创建具有指定名称的索引
flush()	刷新集群中的所有索引
flushAndRefresh()	集合 flush（）和 refresh（
调用	
forceMerge()	等待所有的迁移完成，并强制将集
中的所有索引合并到一个段。	
indexExists(name)	检查给定索引是否存在
admin()	返回用于管理任务的AdminClient
clusterService()	返回集群服务java类
cluster()	返回测试集群类，这将在下一段中进行
明	

test cluster methods （集群测试方法）

InternalTestCluster 类是随机测试中集群功能的核心功能，并允许您配置特定设置或重播某些类型的断，以检查自定义代码的运行反应。

方法名称	描述
ensureAtLeastNumNodes(n)	确保集群中最少
行n个节点	
ensureAtMostNumNodes(n)	确保集群中最多
行n个节点	
getInstance()	从指定节点中获取一个 guice 类
例化实例	
getInstanceFromNode()	从随机节点中获取一个
uice 类实例化实例	
stopRandomNode()	随机停止集群中的节点以
拟中断	
stopCurrentMasterNode()	停止当前主节点，
强制选择新的主节点	
stopRandomNonMaster()	随机停止集群中的非
节点以模拟中断	
buildNode()	创建一个新的 elasticsearch 节点
startNode(settings)	创建并开始一个新的 elastic
earch 节点	

changing node settings（改变节点设置）

如果要确保节点的一定配置，让它们作为EsIntegTestCase的一部分启动，，您可以重写nodeSetting（）方法

```
public class Mytests extends ESIntegTestCase {
    @Override
    protected Settings nodeSettings(int nodeOrdinal) {
        return Settings.builder().put(super.nodeSettings(nodeOrdinal)).put("node.mode", "netwo
k").build();
    }
```

Accessing clients （访问客户端）

为了执行任何操作，您需要使用客户端。您可以使用 ESIntegTestCase.client（）方法来获取随机客机。该客户端可以是 TransportClient 或 NodeClient，通常只要程序开始执行后就无需过多关心。 在nternalTestCluster类中有多种客户端选择方法，可以使用ESIntegTestCase.internalCluster（）方对其进行使用。

方法名称	描述
iterator()	所有可用客户端上的迭代器
masterClient()	返回连接到主节点的客户端
nonMasterClient()	返回未连接到主节点的客户端
getInstanceFromNode()	从随机节点中获取一个uice 类实例化实例
clientNodeClient()	返回在客户机节点运行的客端
client(String nodeName)	将客户端返回给指定点
smartClient()	智能返回客户端

Scoping（范围）

在默认情况下，每个测试组件都使用唯一的集群进行测试。在每一个的测试中都删除了他们之间的索和模板。然而有些时候，您需要为每个测试启动一个新的集群——比如：您想加载某个插件，而不想每一个测试中加载它。

您可以在类级别上使用 @ClusterScope 注释来配置此行为。

```
@ClusterScope(scope=TEST, numDataNodes=1)
public class CustomSuggesterSearchTests extends ESIntegTestCase {
    // ... tests go here
}
```

上述例子为每个测试方法进行配置，配置内容为在测试中使用新的集群。默认范围是 SUITE （测试方中所有方法的一个集群）。numDataNodes设置允许您只启动一定数量的数据节点，这可以加速测执行，因为启动新节点是一项费用高昂且耗时的操作，可能不需要此测试。

默认情况下，测试基本结构将随机启动指定主节点。如果要禁用指定主节点，可以用与numDataNods 类似的设置方式设置 supportsDedicatedMasters = false。如果不使用指定主节点，普通数据节也将被允许成为主节点。

Changing plugins via configuration（通过配置更改插件）

当 elasticsearch 使用 JUnit 4 时，使用 @Before 和 @After 注释时是没有问题的。然而，您应该记，这点在集群设置中没有任何影响，因为当运行这些方法时，集群已经处于启动并运行的状态。所以您如果想在节点运行前配置这些设置，如：在节点启动前载入插件，您应该重写 ESIntegCase这个类的nodePlugins() 方法，并返回每个节点应加载的插件类。

```
@Override
protected Collection<Class<? extends Plugin>> nodePlugins() {
  return Arrays.asList(CustomSuggesterPlugin.class);
```

Randomized testing（随机测试）

Randomized testing（随机测试）

您目前看到的代码片段都没有显示随机测试功能，因为它们被小心隐藏在hood下。 但是，当您编写己的测试时，您也应该使用这些功能。 在开始之前，您应该要知道，如何重复使用相同设置的失败测，如何失败的。 幸运的是，这很容易，因为整个mvn调用与失败的测试都将一起记录，这意味着你以简单地复制和粘贴该行并运行测试。

generating random data（生成随机数据）

下一步是把使用静态测试数据的测试，转换为使用随机测试数据的测试。您可以随机化的数据类型与正在测试的功能有很大的不同。看看下面的例子（注意，这个列表可以继续页面，因为分布式系统有多很多移动的部分）：

- 用任意UTF8符号搜索数据
- 每次运行时更改映射配置，索引和字段名称
- 每次运行时更改响应大小/可配置限制
- 创建索引时更改碎片/副本的数量

那么，你如何创建随机数据。最重要的事情是，你不应该实例化自己的随机实例，而是使用随机化测中提供的随机实例，所有弹性搜索依赖测试类都从该实例继承。

方法名称	描述
getRandom()	返回随机实例，可以在使用特定参调用测试时重新创建
randomBoolean()	返回随机布尔值
randomByte()	返回一个随机字节
randomShort()	返回随机短整数
randomInt()	返回一个随机整数
randomLong()	返回一个随机长
randomFloat()	返回随机浮点数
randomDouble()	返回一个随机的double
randomInt(max)	返回0和max之间的随机整数

between()	返回提供的范围之间的随机值
atLeast()	返回至少为指定整数的随机整数
atMost()	返回最多指定整数的随机整数
randomLocale()	返回随机区域设置
randomTimeZone()	返回随机时区
randomFrom()	从列表/数组返回一个随机元素

此外，还有几种帮助方法，可以创建随机ASCII和Unicode字符串，请参阅随机测试类中以randomAsci，randomUnicode和randomRealisticUnicode开头的方法。后者尝试通过非任意随机创建更真实的nicode字符串。

如果要调试特定随机种子的特定问题，可以使用@Seed注释配置特定种子进行测试。如果要多次运测试，而不是重复启动整个测试套件，可以使用@Repeat注释和任意值。每次迭代都不同于使用不的种子运行。

Assertions(断言)

Assertions

因为许多elasticsearch测试对类似的输出进行检查，比如命中的次数、第一次命中或特殊的高亮显示已经创建了一些预定义的断言。这些已经被放到了弹性搜索断言类中。还有一个特定地理ElasticsearcGeoAssertions断言

方法名	描述
assertHitCount()	检查一个cout请或查询的结果中数量
assertAcked()	确保请已经被主节点确认
assertSearchHits()	断言一个查询结果中包含指定ds
assertMatchCount()	断言一个过滤结果匹配某数量
assertFirstHit()	断言第一次查询命中结果与指定匹配器匹配
assertSecondHit()	断言第二次查询命中结果与定的匹配器匹配
assertThirdHit()	断言第三次查询命中结果与指的匹配器匹配
assertSearchHit()	断言指定的匹配器的查询结果含确定元素
assertNoFailures()	断言没有分片失败的情况
assertFailures()	断言有分片在查询过程中失败的况
assertHighlight()	断言与指定的高亮匹配
assertSuggestion()	断言指定的建议

assertSuggestionSize()	断言指定的建议结果大小
assertThrows()	断言抛出某个异常

Common matchers(常用的匹配器)

-

方法名	描述
hasId()	是否匹配某个ID
hasType(	是否匹配某个类型
hasIndex()	是否匹配某个索引
hasScore()	是否匹配某个确切的评分值
hasStatus()	匹配指定的集群健康状态

通常,你应该像如下组合断言和匹配器

```
SearchResponse searchResponse = client().prepareSearch() ...;
assertHitCount(searchResponse, 4);
assertFirstHit(searchResponse, hasId("4"));
assertSearchHits(searchResponse, "1", "2", "3", "4");
```