## Elasticsearch 6.X 插件开发指南（一）

#### 插件简介

用户通过编写符合一定规则的插件可以使Elasticsearch实现更符合需求的功能，包括自定义映射类型、自定义分析器、原生脚本、自定义发现等等。插件包含JAR文件，但也可能包含脚本和配置文件，并且必须安装在集群中的每个节点上。安装之后，必须重新启动每个节点，插件方可使用。

**注意：具有自定义集群状态元数据（比如X-pack）的插件安装要求集群整个重启。仍然可以通过滚动重启来升级这些插件。**

插件的分类：

核心插件

此类插件属于Elasticsearch项目的一部分。与Elasticsearch同时发布的，它们的版本号总是与Elasticsearch本身的版本号匹配。这些插件由Elastic团队在优秀社区成员的帮助下维护(对于开源插件)。问题和bug报告在Github项目页（https://github.com/elastic/elasticsearch）。

社区贡献插件

此类插件不属于Elasticsearch项目的一部分。它们由独立的开发人员或公司提供，并且拥有自己的许可证和版本控制。问题和bug报告通常可以在社区插件的网站上报告。

#### **官方插件帮助手册**

Elasticsearch源码库包含以下示例：(https://github.com/elastic/elasticsearch/tree/master/plugins/examples）:

* 一个包含自定义设置插件的Java插件(customsettings)。
* 一个Java插件，它包含一个注册Rest处理程序的插件(rest-handler)。
* 一个包含rescore插件的Java插件([rescore](https://github.com/elastic/elasticsearch/tree/master/plugins/examples/rescore" \o "rescore))。
* 一个包含脚本插件的Java插件([script-expert-scoring](https://github.com/elastic/elasticsearch/tree/master/plugins/examples/script-expert-scoring" \o "script-expert-scoring))。

这些例子提供了入门所需的基本知识。关于如何编写插件的更多信息，我们建议查看本文档中列出的插件以获得灵感。

#### **插件描述文件**

插件必须包含具有特定名字的描述文件，文件名为plugin-descriptor.properties。文件格式的细节见下面的例子。

# Elasticsearch插件描述文件

# 插件中必须包含“plugin-descriptor.properties”这个文件

#

### "foo"例子插件

#

# foo.zip <-- 具有下列结构的插件zip包:

# |\_\_\_\_ <随意命名>.jar <-- 类，资源，依赖包

# |\_\_\_\_ <随意命名\_N>.jar <-- 任意数量的jar包

# |\_\_\_\_ plugin-descriptor.properties <-- 文件内容例子如下

#

# classname=foo.bar.BazPlugin

# description=My cool plugin

# version=6.0

# elasticsearch.version=6.0

# java.version=1.8

#

### 所有插件必须包含下列元素:

#

# 'description': 插件的简单摘要

description=${description}

#

# 'version': 插件版本号

version=${version}

#

# 'name': 插件名字

name=${name}

#

# 'classname':加载类名，完全限定（ fully-qualified）

classname=${classname}

#

# 'java.version': 代码开发使用的java版本

#使用系统属性java.specific .version

#版本字符串必须是非负的十进制整数列

#用“.”隔开并且可能以0开头

java.version=${javaVersion}

#

# 'elasticsearch.version': 编译的elasticsearch版本elasticsearch.version=${elasticsearchVersion}

### 以下为插件的可选元素:

#

# 'extended.plugins': 通过SPI此插件扩展的其他插件extended.plugins=${extendedPlugins}

#

# 'has.native.controller':插件是否有本地控制器 has.native.controller=${hasNativeController}

您可以自己填写这个模板，或者，如果您正在使用Elasticsearch的Gradle构建系统，您可以在build.gradle文件填写必要的值。

注意，只有插件根目录下的jar文件才会被添加到插件的类路径中!如果您需要其他资源，请将它们打包到资源jar中。

**重要：插件发布生命周期**

你必须为每一个新的Elasticsearch版本发布一个新版本的插件。插件加载时进行版本检查，因此如果插件的Elasticsearch .version不正确，则Elasticsearch将拒绝启动。

#### **插件测试**

在测试Java插件时，只有在plugins/目录中才会自动加载。使用命令bin/elasticsearch-plugin install [file:///path/to/your/plugin](file://path\\to\\your\\plugin) 安装你的插件进行测试。您还可以在集成测试的测试框架中加载插件。

#### **JAVA安全权限**

如果插件需要额外的安全权限，需要在plugin-security.policy文件通过grant声明额外的权限。任何额外的权限都会显示给用户，并带有一个很大的警告，用户必须在交互式安装插件时确认这些权限。因此，如果可能的话，最好避免请求任何虚假的权限!

如果您正在使用Elasticsearch Gradle构建系统，请将该文件放在src/main/plugin-metadata中，它也将在单元测试期间应用。

请记住，Java安全模型是基于堆栈的，额外的权限只授予插件中的jar，因此您必须围绕需要提升特权的操作编写适当的安全代码。建议添加一个检查，以防止非特权代码(如脚本)获得升级权限。例如:

//在doPrivileged()代码块之前必须检查ES权限

import org.elasticsearch.SpecialPermission;

SecurityManager sm = System.getSecurityManager();

if (sm != null) {

//没有特权的代码，比如脚本，没有SpecialPermission

sm.checkPermission(new SpecialPermission());

}

AccessController.doPrivileged(

// 敏感操作

);

## Elasticsearch 6.X 插件开发指南（二）

看完Elasticsearch插件的官方帮助文档，相信你一头雾水，me too！别着急，接下来，我们看看如何真正开发一个插件。（一）中介绍了插件的功能，即可以向Elasticsearch添加新功能，而不必创建Elasticsearch本身的分支。在本文中，我们将从头开始介绍创建一个新的Elasticsearch插件所需的步骤。

#### **建立项目构建结构**

开发Elasticsearch插件的唯一要求是:

1. 可运行的Maven
2. Java编译器

打包的插件只是一个Zip文件，包含一个或多个Java Jar文件，其中包含已编译的代码和资源。一旦插件被编写和打包，可以很容易地通过一行命令将其添加到任何Elasticsearch中。命令如下：bin/elasticsearch-plugin install [file:///path/to/your/plugin](file://path\\to\\your\\plugin)

首先为示例插件创建必要的插件结构:

$ mkdir example-plugin

$ cd example-plugin

$mkdir -p src/main/{java,resources,assemblies}

$mkdir -p src/main/java/org/elasticsearch/plugin/example

接下来，我们需要一个Maven配置文件，按照惯例，这个配置文件将写入pom.xml，为节约篇幅，较长代码不再粘贴，请看https://github.com/NikolaDai/Elasticsearch-plugin-development/blob/master/example-plugin/pom.xml。

虽然上面Maven配置文件的大部分都是样板文件，定义了一些必需的属性并配置了依赖项，但是本例中有趣的部分是对src/main/ assembly /plugin.xml的引用，在其中我们定义了插件的打包方式。

#### **让Elasticsearch知道我们的插件**

虽然以上两个文件足以让插件构建过程正常工作，但是我们需要告诉Elasticsearch关于插件的信息，这是通过将插件的名称添加到类路径上一个特殊的es-plugin.properites文件来完成的。我们通过创建src/main/resourc/es-plugin.properties来实现:

plugin=org.elasticsearch.plugin.example.ExamplePlugin

当Elasticsearch启动时，使用Elasticsearch org.elasticsearch.plugins。PluginManager将扫描当前的类路径以查找插件配置文件，并实例化引用的插件。es-plugin.properties文件是一个简单的Java属性文件，它必须包含关键值plugin。该值应该是兼容插件类的完全限定类名。

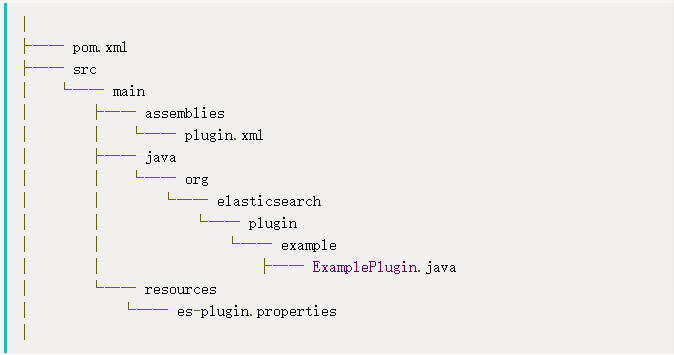
一个简单的插件扩展org.elasticsearch.plugins.AbstractPlugin，它减少了样板代码，看起来像这样:

package org.elasticsearch.plugin.example;  
  
import org.elasticsearch.plugins.AbstractPlugin;  
public class ExamplePlugin extends AbstractPlugin {  
 public String name() { return "example-plugin"; }  
 public String description() { return "Example Plugin Description"; }  
}

在Elasticsearch中使用这个名称来标识插件，例如在打印加载的插件列表时。该描述目前没有用于任何用途，但可能用于未来潜在的插件管理或信息API。

#### **插件目录树结构**

我们刚刚创建的四个文件实际上足以创建一个Elasticsearch插件。如果您遵循上述步骤，您的目录树应该是这样的:



由于基于maven，大多数Java ide(如IDEA和Eclipse)都能够从pom中读取项目定义, 并自动配置项目和下载所需的依赖项。 在下一节中，我们将展示如何从命令行构建插件，以及如何安装插件。

#### **构建和安装插件**

在发布和使用插件之前，它必须被组装，这是通过Maven完成的:

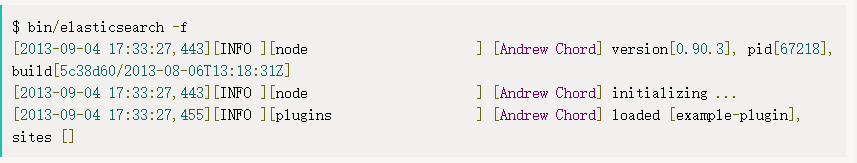
$ mvn package

上面的命令将我们的插件包组装成一个Zip文件，可以使用Elasticsearch插件命令来安装:

$ bin/plugin --url file:///PATH-TO-EXAMPLE-PLUGIN/target/releases/example-plugin-1.0-SNAPSHOT.zip --install example-plugin

注意，我们使用插件命令的—url选项是为了通知它在本地获取文件，而不是试图从在线存储库下载文件。

我们现在可以开始Elasticsearch，看看我们加载的插件:



然而，由于我们的插件实际上并没有做任何事情，很少有证据表明我们已经完成了一些事情。下一节将演示如何向内置HTTP服务器返回简单的HTTP响应。

#### **例子：说Hello**

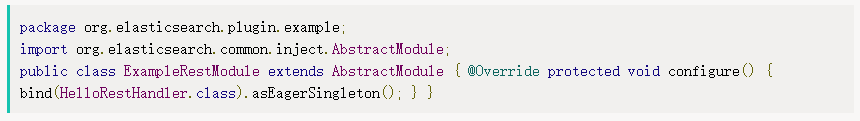
插件可以以多种方式为Elasticsearch添加功能。在本节中，我们将展示如何使用Guice向Elasticsearch添加模块。在Elasticsearch中使用Guice将服务器的所有组件连接在一起，并为服务器提供文档化良好的入口点。描述Guice超出了本文的范围，感兴趣的读者应该可以毫不费力地找到关于Guice框架的多个教程和介绍。

#### **添加新模块**

要添加一个新的模块Elasticsearch，我们覆盖AbstractPlugin上的abstract modules()方法，该方法允许我们通过返回对集合中其类的引用来添加多个模块。我们开始添加一个新的模块到我们的ExamplePlugin:



在初始化期间，Elasticsearch实例化这些模块，并调用它们的configure()方法。在这个方法中，我们必须确保RestHandler被添加到要实例化的类列表中:



通过将类作为一个独立的实例绑定到自身，我们通知框架我们希望它只实例化HelloRestHandler类的一个实例。一旦构造器所有参数都被实例化后，用@Inject注释的构造器就会被调用来实例化这个类。

#### **创建HTTP处理程序**

在Elasticsearch中，所有REST请求都由org.elasticsearch.rest.RestController实例来处理。它维护到org.elasticsearch.rest.RestHandler实例的路径的内部路由，并依次处理请求。

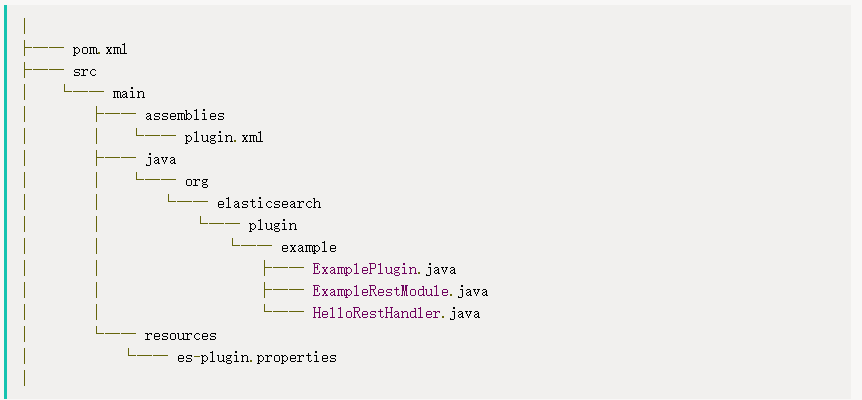
添加对新HTTP请求的支持就像通过Guice获得对RestController的引用并向控制器注册处理程序一样简单。

我们的HelloRestHandler应该看起来像这样：



#### **构建和安装更新后的插件**

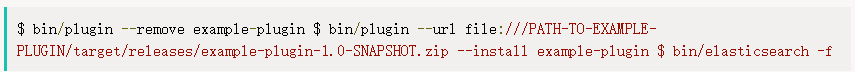
更新后的目录树结构如下：



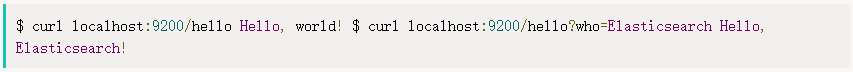
我们可以继续重新打包我们的插件，更新Elasticsearch并验证插件是否工作。使用Maven在example-plugin文件夹中构建包:

$ mvn package

要更新Elasticsearch中安装的插件，我们必须停止Elasticsearch，卸载插件，重新安装插件，并重新启动Elasticsearch:



使用另一个终端，我们现在可以确认插件正常工作:



#### **结论**

在本文中，我们展示了如何从头创建一个新的Elasticsearch插件。虽然开始编写用于Elasticsearch的插件相对容易，但是编写更复杂的插件通常需要深入了解当前Elasticsearch代码库。

更新:在pom.xml中添加< appendassemyid >false</appendAssemblyId>以排除文件名中的-plugin后缀。当将插件发布到Maven存储库时，这种方法效果更好。

## Elasticsearch 6.X 插件开发指南（三）