Elasticsearch-1.6.0

监控&插件&源码调试

王志

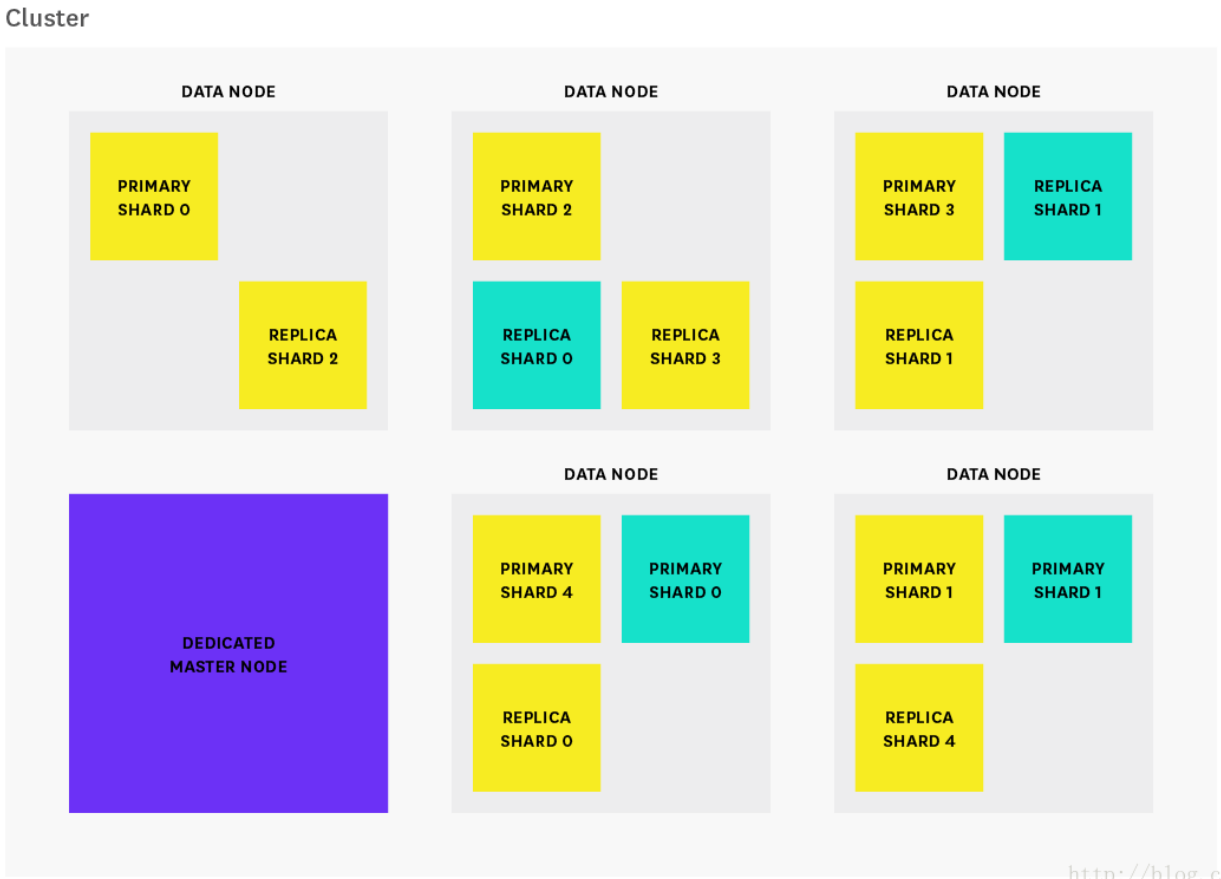
2017.12

1. Elasticsearch脚本监控

2. Elasticsearch 插件介绍

3. Elasticsearch 源码调试

Elasticsearch 监控



- elasticsearch cluster （集群整体状态）
- Elasticsearch node （集群节点状态）

Elasticsearch监控

- Cluster的监控（localhost:9200）
  1. Cluster Health -查看集群健康状态接口 （ /_cluster/health）
  2. Cluster State -查看集群状况接口 （ /_cluster/state）
  3. Cluster Stats -查看集群统计信息接口（ /_cluster/stats）
  4. Pending cluster tasksedit – 查看集群任务（ _cluster/pending_tasks ）

- Nodes层面的监控（localhost:9200）
  1. Nodes Stats -节点状态（ /_nodes/stats）
  2. Nodes Info -节点信息（/_nodes ）
  3. Nodes hot_threads - 节点的热线程(/_nodes/hot_threads)

Elasticsearch监控

- **Elasticsearch 性能监控指标**
  1. Search and indexing performance - 搜索性能指标
  2. Memory and garbage collection - 索引性能指标
  3. Host-level system and network metrics - 内存使用和GC指标
  4. Cluster health and node availability - 集群健康和节点可用性
  5. Resource saturation and errors - elasticsearch主机的网络和系统
  6. Resource saturation and errors - 资源saturation and errors

参考链接： http://www.jianshu.com/p/6574e2288745

Elasticsearch 监控

- **搜索性能指标的要点**
  1. Query load： 监控当前请求数量
  2. Query latency: query阶段时延（总耗时/总查询次数）
  3. Fetch latency: 提取阶段平均时延

Metric description	Name	[Metric type]
Total number of queries	indices.search.query_total	Work: Throughput
Total time spent on queries	indices.search.query_time_in_millis	Work: Performance
Number of queries currently in progress	indices.search.query_current	Work: Throughput
Total number of fetches	indices.search.fetch_total	Work: Throughput
Total time spent on fetches	indices.search.fetch_time_in_millis	Work: Performance
Number of fetches currently in progress	indices.search.fetch_current	Work: Throughput

Elasticsearch 监控查询

- **索引性能指标**
  1. Indexing latency :监控index的平均耗时
  2. Refresh latency：监控refresh的平均耗时
  3. Flush latency：监控flush的平均耗时

Metric description	Name	[Metric type]
Total number of documents indexed	indices.indexing.index_total	Work: Throughput
Total time spent indexing documents	indices.indexing.index_time_in_millis	Work: Performance
Number of documents currently being indexed	indices.indexing.index_current	Work: Throughput
Total number of index refreshes	indices.refresh.total	Work: Throughput
Total time spent refreshing indices	indices.refresh.total_time_in_millis	Work: Performance
Total number of index flushes to disk	indices.flush.total	Work: Throughput
Total time spent on flushing indices to disk	indices.flush.total_time_in_millis	Work: Performance

Elasticsearch 监控查询

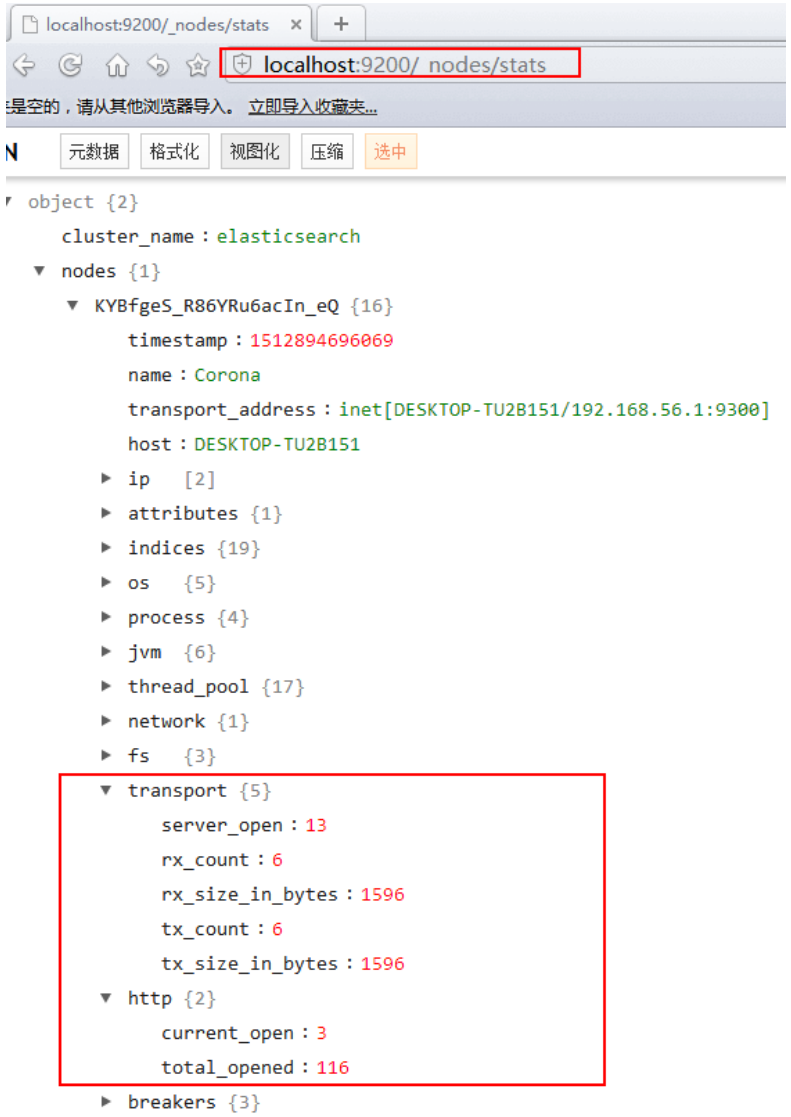
- **内存使用和GC指标**
  1. Garbage collection duration and frequency
  2. JVM heap in use、JVM heap used、JVM heap committed、Memory usage

Metric description	Name	[Metric type]
Total count of young-generation garbage collections	jvm.gc.collectors.young.collection_count	Other
Total time spent on young-generation garbage collections	jvm.gc.collectors.young.collection_time_in_millis	Other
Total count of old-generation garbage collections	jvm.gc.collectors.old.collection_count	Other
Total time spent on old-generation garbage collections	jvm.gc.collectors.old.collection_time_in_millis	Other
Percent of JVM heap currently in use	jvm.mem.heap_used_percent	Resource: Utilization
Amount of JVM heap committed	jvm.mem.heap_committed_in_bytes	Resource: Utilization

Elasticsearch 监控查询

- **elasticsearch主机的网络和系统**
  1. Disk space
  2. I/O utilization
  3. CPU utilization
  4. Network bytes sent/received
  5. Open file descriptors
  6. HTTP connections

Name	**[Metric type]**	
Available disk space	Resource: Utilization	
I/O utilization	Resource: Utilization	
CPU usage	Resource: Utilization	
Network bytes sent/received	Resource: Utilization	
Open file descriptors	Resource: Utilization	
Metric description	**Name**	**[Metric type]**
Number of HTTP connections currently open	http.current_open	Resource: Utilization
Total number of HTTP connections opened over time	http.total_opened	Resource: Utilization

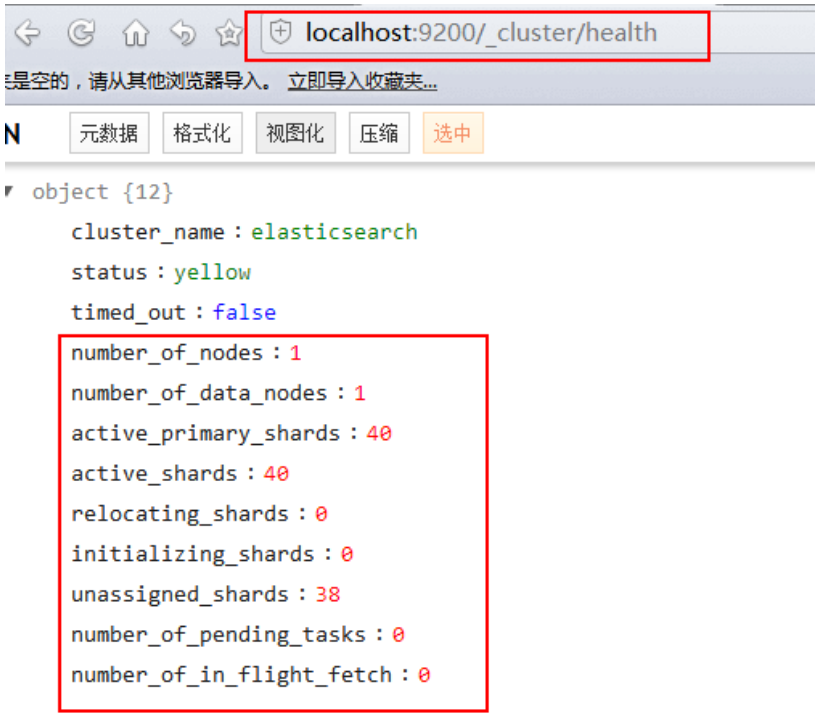


Elasticsearch 监控查询

集群健康和节点可用性

1. Cluster status: 黄色代表副本分片未分配或丢失，红色代表主分片丢失。
2. Initializing and unassigned shards：分片仍处于初始化或未分配状态

Metric description	Name	[Metric type]
Cluster status (green, yellow, red)	cluster.health.status	Other
Number of nodes	cluster.health.number_of_nodes	Resource: Availability
Number of initializing shards	cluster.health.initializing_shards	Resource: Availability
Number of unassigned shards	cluster.health.unassigned_shards	Resource: Availability

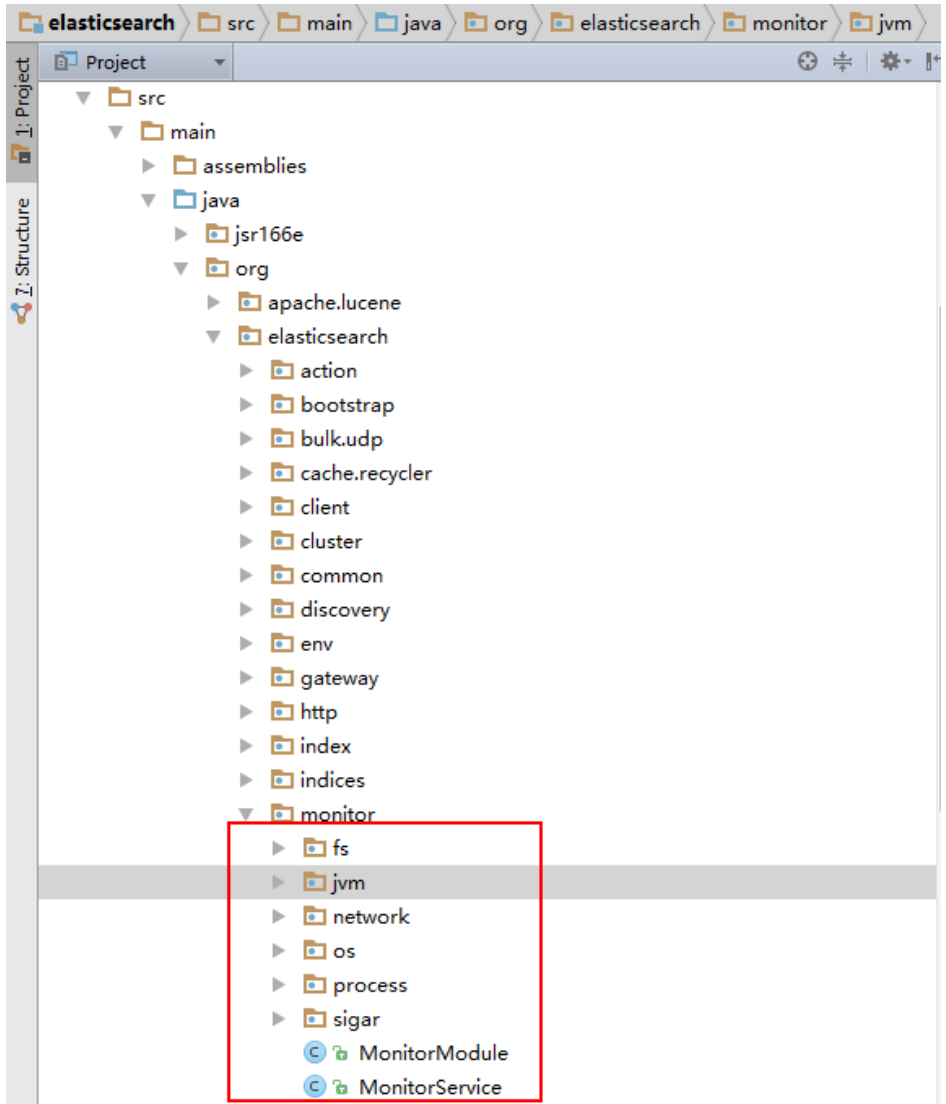


Elasticsearch 监控查询

- 监控源码探究
  1. JMX
  2. Sigar

- 监控数据采集
  1. 文档见附件
  2. Demo：github

Microsoft Word
文档



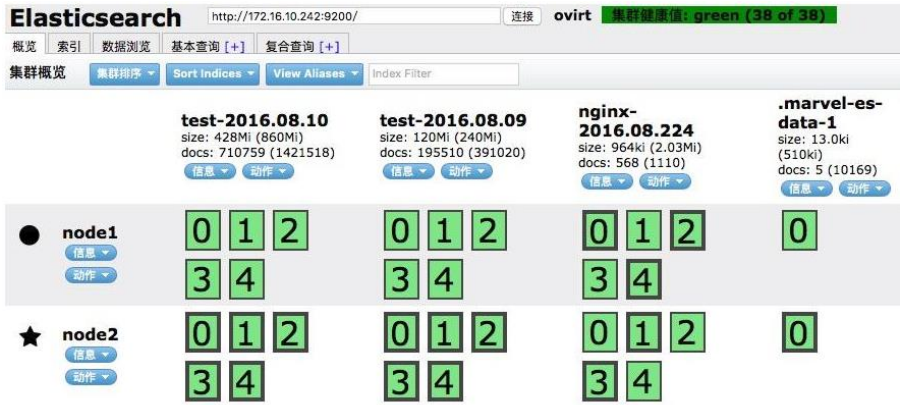
1. Elasticsearch脚本监控

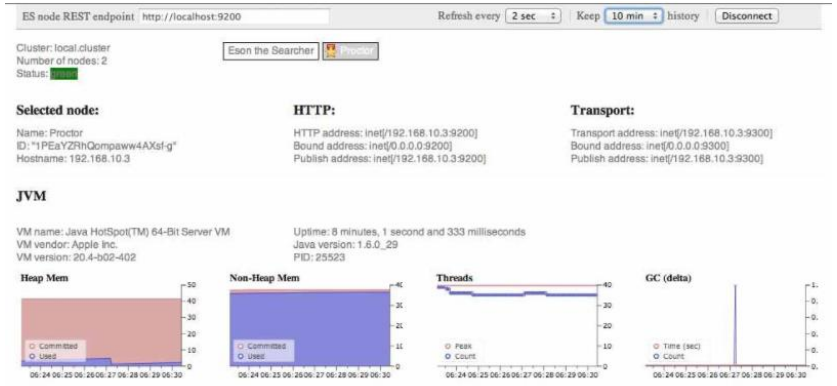
2. Elasticsearch 插件介绍

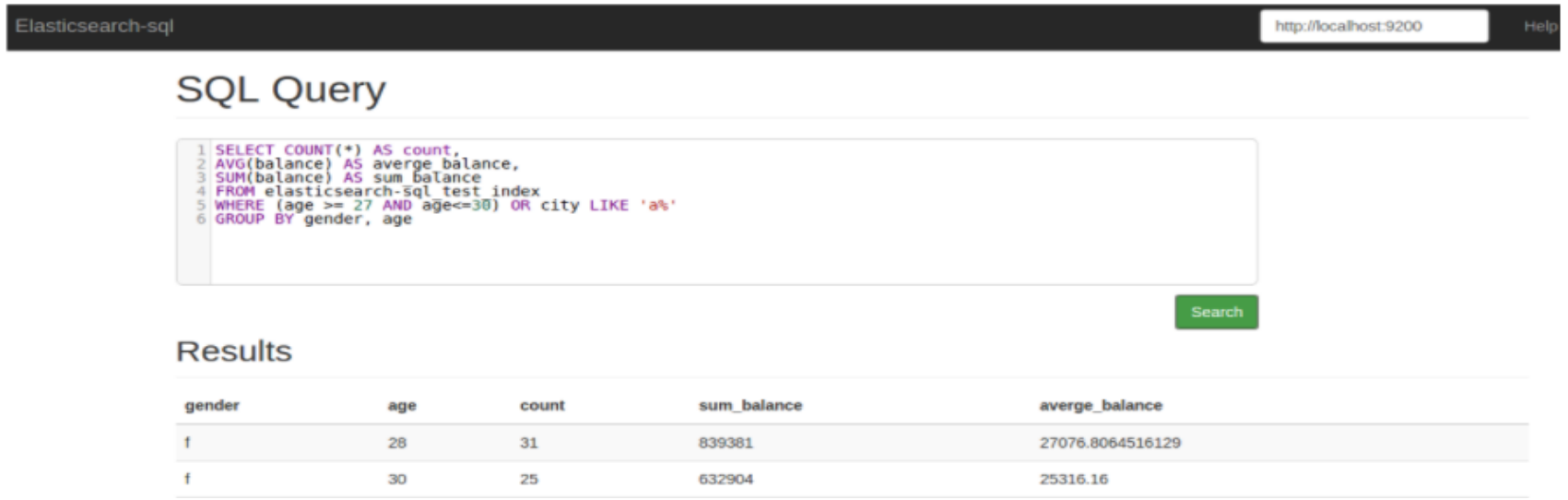
3. Elasticsearch 源码调试

Elasticsearch 插件介绍

- 插件介绍
  1. Head
  2. Bigdeck
  3. Sql
  4. more

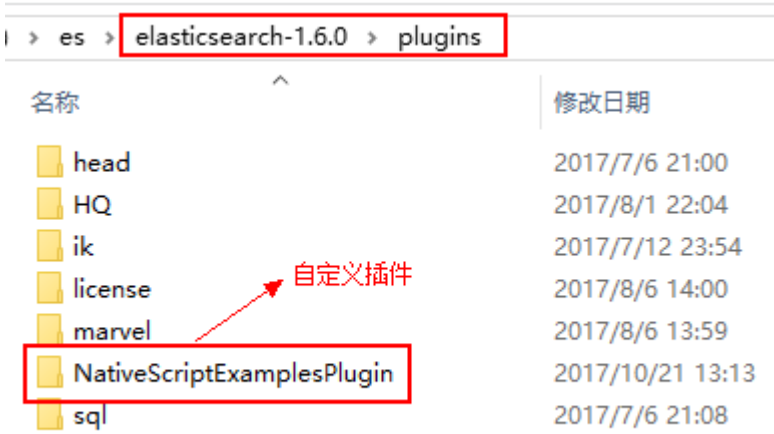






Elasticsearch 插件介绍

- 插件安装
  1. 自动安装：
     1. 在Elasticsearch目录下
     2. $/bin/plugin -install mobz/elasticsearch-head
  2. 手动安装
     1. 在elasticsearch-1.6.0\plugins新建目录
     2. 拷贝自定义插件的jar包到目录下
     3. Es重启自动加载插件
  3. 插件查看
     1. http://localhost:9200/_cat/plugins



Elasticsearch 插件介绍

- 插件开发
  1. 打分(script)插件：用于自定义打分
  2. Rest插件：用于浏览器访问
  3. Cluster插件：用于操作集群所有节点

- Demo
  - https://github.com/lebron374/elasticsearch_cluster_operation

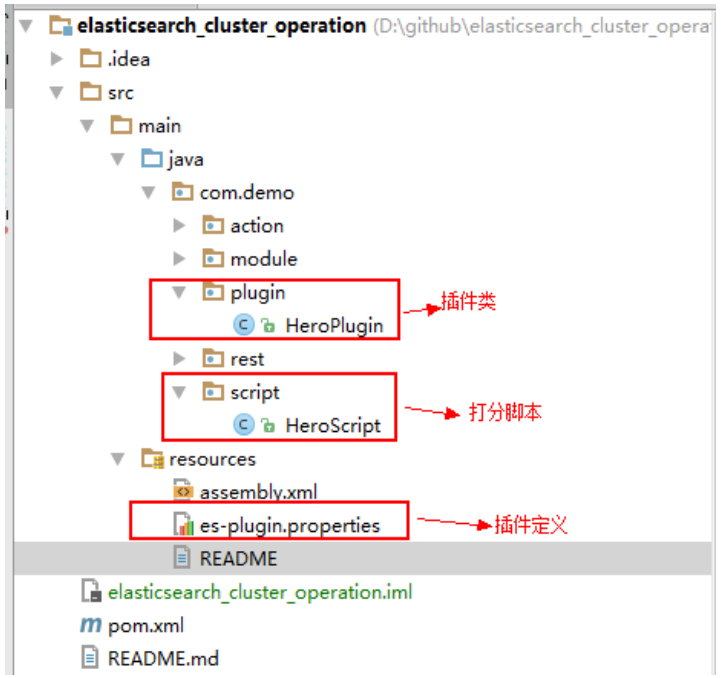
- 参考资料（doc文档为插件加载过程）
  - http://www.jianshu.com/p/b32b838d5220
  - http://www.jianshu.com/p/17d4477c0954

Elasticsearch 插件介绍

- 插件开发
  - 定义Plugin插件类
  - 定义打分script类并通过plugin类注册
  - 新增es-es-plugin.properties
  - 打包编译生成jar包

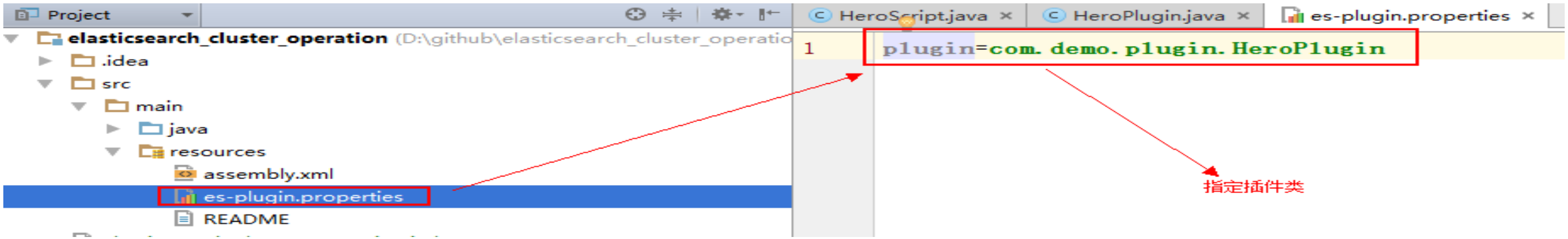
- 参考资料
  - 参考文档：http://www.jianshu.com/p/7a9af40a29f7
  - Demo：https://github.com/lebron374/elasticsearch_cluster_operation
  - Demo：https://github.com/lebron374/elasticsearch_rest_plugin_demo

Elasticsearch - 打分插件

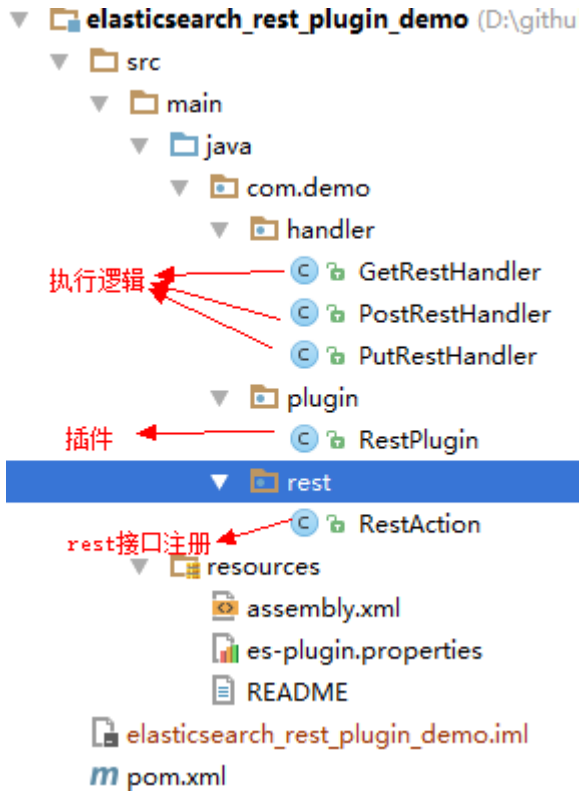



```
public class HeroScript implements NativeScriptFactory {

    public static final String SCRIPT_NAME = "hero_script";
    public Map<String, Object> localParams; // 在针对每个doc执行算分的时候可以访问传递下来的参数

    /**
     * 创建执行脚本对象，在一次query当中只会生成一个对象
     * @param params score query 传下来的参数
     * @return
     */
    @Override
    public ExecutableScript newScript(@Nullable Map<String, Object> params) {
        localParams = params;

        return new HeroActionScript();
    }

    /**
     * 实现打分的逻辑类
     */
    private class HeroActionScript extends AbstractDoubleSearchScript {

        /**
         * 真正执行打分部分的逻辑，可以获取doc的信息
         * @return 算法
         */
        @Override
        public double runAsDouble() {
            //执行一些打分逻辑,打分的时候可以获取传递下来的参数，可以获取doc文档的field字段以及更多，可以通过doc()跟进查看源码
            System.out.println(localParams.toString());
            System.out.println(this.doc().toString());
            return 0;
        }
    }
}
```

```
public class HeroPlugin extends AbstractPlugin {

    public String name() { return "hero"; }

    public String description() { return "hero"; }

    /**
     * onModule接口在InternalNode.setup()接口中初始化
     * onModule接口在InternalNode.start中的injector = modules.createInjector();进行初始化
     * @param module
     */
    public void onModule(ActionModule module) {
        module.registerAction(HeroAction.INSTANCE, TransportNodesHeroAction.class);
    }

    public void onModule(RestModule module) {
    }

    public void onModule(ScriptModule module) {
        module.registerScript(HeroScript.SCRIPT_NAME, HeroScript.class);
    }
}
```



Elasticsearch - rest插件




```
public class RestAction extends BaseRestHandler{

    @Iniect
    public RestAction(Settings settings, Client client, RestController controller) {
        super(settings, controller, client);

        controller.registerHandler(RestRequest.Method.GET, "/rest", new GetRestHandler(client));
        controller.registerHandler(RestRequest.Method.POST, "/rest", new PostRestHandler(client));
        controller.registerHandler(RestRequest.Method.PUT, "/rest", new PutRestHandler(client));
    }

    @Override
    protected void handleRequest(RestRequest request, RestChannel channel, Client client) throws E


    }
}
```

注册接口

```
public class RestPlugin extends AbstractPlugin {

    @Override
    public String name() { return "rest demo"; }

    @Override
    public String description() { return "rest demo"; }

    /**
     *
     * @param module
     */
    public void onModule(RestModule module) { module.addRestAction(RestAction.class); }

}
```

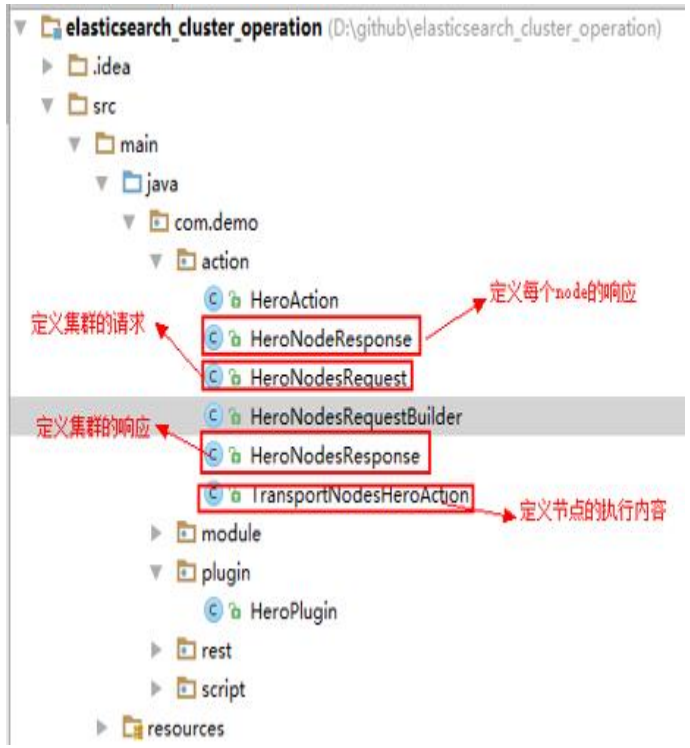
注册插件

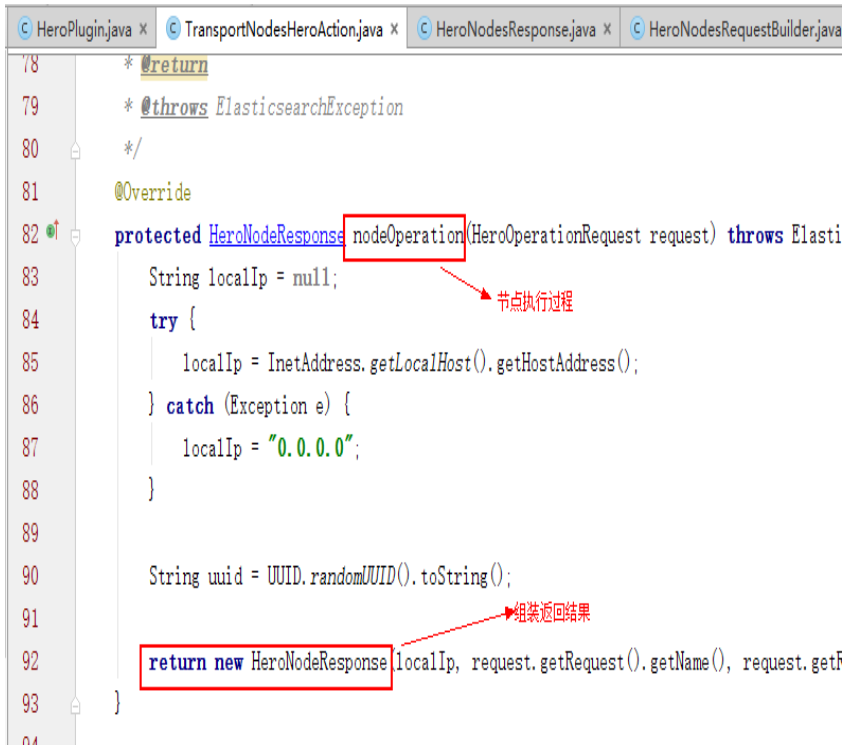
```
public class GetRestHandler implements RestHandler {

    private final ESLogger esLogger = ESLoggerFactory.getLogger(GetRestHandler.class.getName());

    private Client client;
    private String prefix;

    public GetRestHandler(Client client) { this.client = client; }

    @Override
    public void handleRequest(RestRequest request, RestChannel channel) throws Exception {

        /* 获取参数的方法 */
        prefix = request.param("prefix");

        /* 执行具体的动作 */
        NodesInfoResponse response = client.admin().cluster().prepareNodesInfo().all().execute().actionGet();
        List<String> nodes = new ArrayList<>();
        for (NodeInfo nodeInfo : response.getNodes()) {
            String nodeName = nodeInfo.getNode().getName();
            nodes.add(nodeName);
        }

        /* 回传数据 */
        try {
            sendResponse(request, channel, nodes);
        } catch (IOException ioe) {
            esLogger.error("Error sending response", ioe);
        }
    }

    private void sendResponse(RestRequest request, RestChannel channel, List nodes) throws IOException {
```

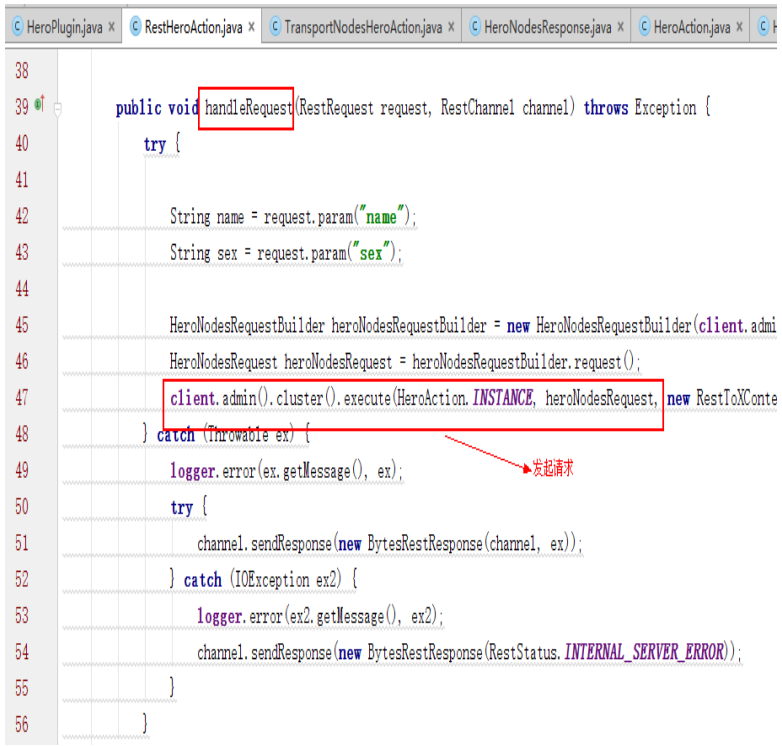
处理请求

返回响应

Elasticsearch – cluster插件







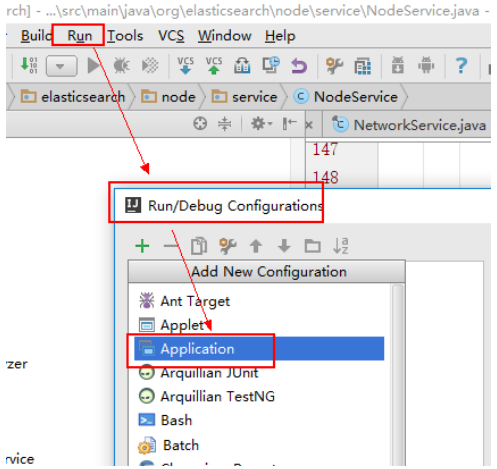
1. Elasticsearch脚本监控

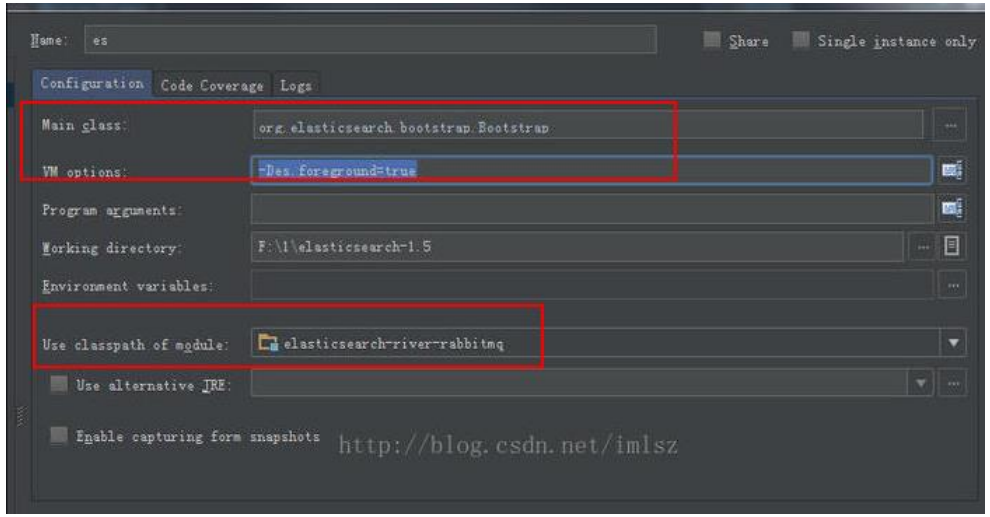
2. Elasticsearch 插件介绍

3. Elasticsearch 源码调试

Elasticsearch 源码调试

- 源码调试步骤
  1. 导入elasticsearh源码
  2. 导入自定义插件的源码
  3. Idea上按照右图进行配置





- 参考资料

  http://www.jianshu.com/p/83d49e3e341b

Elasticsearch 源码参考

- 参考资料
  1. 源码阅读： http://www.jianshu.com/p/e84f868335b3
  2. 插件介绍： https://www.cnblogs.com/huangfox/p/3541300.html
  3. 自定义插件： http://www.jianshu.com/p/7a9af40a29f7
  4. Plugin demo：https://github.com/lebron374/elasticsearch_cluster_operation
     https://github.com/lebron374/elasticsearch_rest_plugin_demo
  5. Es性能监控：http://www.jianshu.com/p/6574e2288745