cloudera®

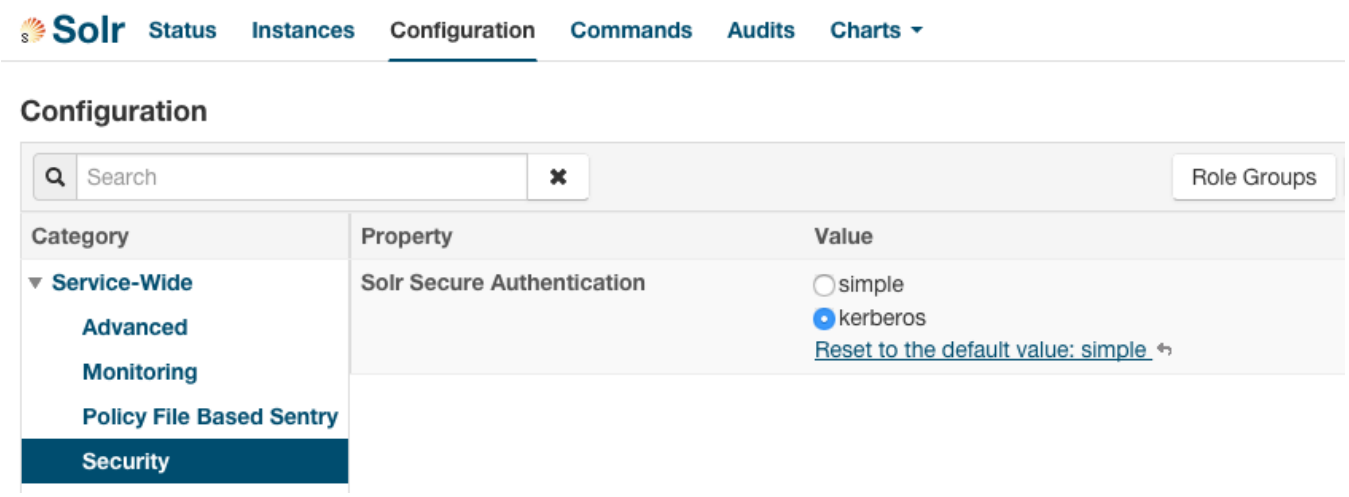
Cloudera Search 安全配置案例

Cloudera Search 作为Hadoop 平台内置的搜索引擎，使得更多的用户，包括数据分析师、管理人员等，体验大数据的优势。可是如何保证机密信息在方便使用的同时保证安全可控？如何确保用户只能访问其限制级以内的信息？

前提要求：ing

1. CDH 集群已经开启 Kerberos 认证功能
2. 使用 kadmin 创建用户 dol (全名为: dol@CDH.COM)

通过 CDH 配置向导添加 Solr 服务，默认情况下，采用 Simple 的认证方式。修改为 Kerberos，重新启动 Solr 服务：



重启后，确认对应服务的 principal 是否已经建立，在 KDC 所在机器运行以下命令：

kadmin.local -q 'list_principals' | grepsolr

如果正确运行，至少可以看到一个 principal，例如:

```
[root@ip-172-31-4-47 ~]# kadmin.local -q 'list_principals' | grep solr
solr/ip-172-31-4-47.us-west-1.compute.internal@CDH.COM
```

注: 如果 solr 部署到 N 台服务器上，应有 N 个 principal

测试 KerberosSolr

编写 solr collection 的 schema 文件，指定待索引文档的格式

```
……
<field name="id" type="string" indexed="true" stored="true" required="true" multiValued="false"/>
<field name="doc_name" type="string" indexed="true" stored="true"/>
<field name="doc_type" type="string" indexed="true" stored="true"/>
<field name="doc_text" type="text_ws" indexed="true" stored="true"/>
<field name="_version_" type="long" indexed="true" stored="true"/>
……
```

创建 solr collection

```
#!/bin/sh

# 指定zookeeper所在机器的IP地址
ZK="172.31.4.47"

# 指定solr collection的名字
COLLECTION="test"

BASE=`pwd`

echo"create solr collection"
rm-rftmp/*

# 创建solr collection配置文件模板
solrctl--zk$ZK:2181/solrinstancedir--generatetmp/${COLLECTION}_configs

#使用上述自定义的schema文件
cp template/schema.xml tmp/${COLLECTION}_configs/conf/

#使用CDH发行版内置的solrconfig.xml.secure
mvtmp/${COLLECTION}_configs/conf/solrconfig.xml tmp/${COLLECTION}_configs/conf/solrconfig.xml.bak
mvtmp/${COLLECTION}_configs/conf/solrconfig.xml.securetmp/${COLLECTION}_configs/conf/solrconfig.xml

#将配置文件上传至zookeeper
solrctl--zk$ZK:2181/solrinstancedir--create$COLLECTIONtmp/${COLLECTION}_configs

# 创建solr collection
solrctl--zk$ZK:2181/solr collection--create$COLLECTION-s1-r1

# 查询solr collection
solrctl--zk$ZK:2181/solr collection --list
```

准备测试数据 sample/data.json 文件

```
[{"id":"doc1", "doc_name":"spark", "doc_type":"word", "doc_text":"alex spark"}, {"id":"doc2", "doc_name":"impala", "doc_type":"pdf", "doc_text":"alex impala"}]
```

使用 curl 上传数据文件建立索引

```
curl -i --negotiate -u : 'http://172.31.4.47:8983/solr/test/update/json?commit=true' --data-binary @sample/data.json -H 'Content-type:application/json'
```

使用 curl 进行查询

```
[dol@ip-172-31-4-47 solr]$ curl -i --negotiate -u : 'http://172.31.4.47:8983/solr/test/select?q=*%3A*&wt=json&indent=true'

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie:hadoop.auth="u=dol&p=dol@CDH.COM&t=kerberos-dt&e=1427388617366&s=ex2KejiZwKXE56BEZxxM4C/rgpE="; Path=/; Expires=Thu, 26-Mar-2015 16:50:17 GMT; HttpOnly
Content-Type: text/plain;charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 26 Mar 2015 06:50:17 GMT

{
 "responseHeader":{
  "status":0,
  "QTime":0,
  "params":{
   "indent":"true",
   "q":"*:*",
   "wt":"json"}},
 "response":{"numFound":2,"start":0,"docs":[
   {
    "id":"doc1",
    "doc_name":"spark",
    "doc_type":"word",
    "doc_text":"alex spark",
    "_version_":1496687435120115712},
   {
    "id":"doc2",
    "doc_name":"impala",
    "doc_type":"pdf",
    "doc_text":"alex impala",
    "_version_":1496687435217633280}]
 }}
```

更多细节

在使用 curl 命令时，需要指定参数 --negotiate 选项，表示使用 SPNEGO 协议协商底层认证协议 (在该案例中使用 kerberos)。因此在使用 curl 查询时，如果跟踪 HTTP 消息的元数据，还可以看到如下结果：

```
HTTP/1.1 401 Unauthorized
Server: Apache-Coyote/1.1
WWW-Authenticate: Negotiate
Set-Cookie: hadoop.auth=; Path=/; Expires=Thu, 01-Jan-1970 00:00:00 GMT; HttpOnly
Content-Type: text/html;charset=utf-8
Content-Length: 997
Date: Thu, 26 Mar 2015 06:50:17 GMT
```

另外运行 curl 前，必须使用 kinit 获取 TGT 缓存到 kerberos cache；否则，运行 curl 时将无法获取有效用户凭证。导致请求被拒绝。

使用 solrJ Java Library 访问 solr

修改 pom.xml 引入对应的依赖包

```
<dependency>
<groupId>org.apache.solr</groupId>
<artifactId>solr-solrj</artifactId>
<version>4.4.0-cdh5.3.2</version>
</dependency>
```

编写 java 类 com.cloudera.example.SearchEngine

```
SolrServersolr = new HttpSolrServer(url);# url 指定 solr 位置，比如 http://solr-node:8983/solr/test
SolrQuery query = new SolrQuery();
query.set("q", "*:" + param); # 生成 solr 查询条件
QueryResponse response = null;
try {
        response = solr.query(query);# 发送 HTTP 请求，REST API 调用
    } catch (SolrServerException e) {
throw new RuntimeException(e);
    }
if (response != null) {
SolrDocumentList results = response.getResults();
        for (SolrDocument doc : results) {# 遍历打印所有查询结果
logger.info(doc.toString());
        }
    }
```

编写 JAAS 配置文件

```
Client {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=false
useTicketCache=true# 需要使用 kinit 提前申请 TGT
};
```

执行 java 程序，指定 JAAS 配置文件

```
#!/bin/sh

JAR="target/solrapi-0.4.0-jar-with-dependencies.jar"
JAAS="config/jaas.conf"
CLASS="com.cloudera.example.SearchEngine"
# Need to use FQDN (full qualified domain name) rather than IP address
PARAMS="http://ip-172-31-4-47.us-west-1.compute.internal:8983/solr/test"

java -cp$JAR -Djava.security.auth.login.config=$JAAS $CLASS $PARAMS
```

执行后，应该与 curl 命令获得相同的结果，即查询到 2 个文档，并对每个文档的内容进行打印。

Happy Kerberos'edSolr！