1.代码

package com.org.hive;

import org.apache.hadoop.hive.ql.parse.ASTNode;

import org.apache.hadoop.hive.ql.parse.AbstractSemanticAnalyzerHook;

import org.apache.hadoop.hive.ql.parse.HiveParser;

import org.apache.hadoop.hive.ql.parse.HiveSemanticAnalyzerHookContext;

import org.apache.hadoop.hive.ql.parse.SemanticException;

import org.apache.hadoop.hive.ql.session.SessionState;

public class AuthHook extends AbstractSemanticAnalyzerHook {

private static String admin = "hdfs";

@Override

public ASTNode preAnalyze(HiveSemanticAnalyzerHookContext context,

ASTNode ast) throws SemanticException {

switch (ast.getToken().getType()) {

case HiveParser.TOK\_CREATEDATABASE:

case HiveParser.TOK\_DROPDATABASE:

case HiveParser.TOK\_CREATEROLE:

case HiveParser.TOK\_DROPROLE:

case HiveParser.TOK\_GRANT:

case HiveParser.TOK\_REVOKE:

case HiveParser.TOK\_GRANT\_ROLE:

case HiveParser.TOK\_REVOKE\_ROLE:

String userName = null;

if (SessionState.get() != null

&& SessionState.get().getAuthenticator() != null) {

userName = SessionState.get().getAuthenticator().getUserName();

}

if (!admin.equalsIgnoreCase(userName)) {

throw new SemanticException(userName

+ " can't use ADMIN options, except " + admin + ".");

}

break;

default:

break;

}

return ast;

}

}

2.hive-site中添加配置：

<property>

<name>hive.metastore.authorization.storage.checks</name>

<value>true</value>

</property>

<property>

<name>hive.metastore.execute.setugi</name>

<value>false</value>

</property>

<property>

<name>hive.security.authorization.enabled</name>

<value>true</value>

</property>

<property>

<name>hive.security.authorization.createtable.owner.grants</name>

<value>ALL</value>

</property>

<property>

<name>hive.aux.jars.path</name>

<value>file:///data/hiveAuth/hiveAuth.jar</value>

</property>

<property>

<name>hive.semantic.analyzer.hook</name>

<value>com.org.hive.AuthHook</value>

</property>

3.重启hive服务

4.给用户赋权限：

grant select on database default to user etl;

revoke select on database default to user etl;

GRANT SELECT on TABLE(DATABASE) test1 to USER etl；

grant create on database default to user etl;

grant drop on database default to user etl;

grant update on database default to user etl;

5.给所有用户增加读写执行的权利（注意：非超级管理员用户想要create表必须要执行这一步）

hdfs dfs -chmod =wrx /user/hive/warehouse

因此，如果使用HiverServer2来提供给用户来链接Hive，必须启用安全认证，也就是hive.server2.authentication的配置。

目前HiveServer2支持多种用户安全认证方式：NONE,NOSASL, KERBEROS, LDAP, PAM ,CUSTOM等等。

本文介绍使用自定义的用户认证方式，即CUSTOM；  
如果将hive.server2.authentication设置成CUSTOM，则需要设置  
hive.server2.custom.authentication.class来指定用于权限认证的类，这个类需要实现  
org.apache.hive.service.auth.PasswdAuthenticationProvider接口。

我们将使用HiveServer2的用户名和密码保存起来，其中，密码以32位小写md5加密来保存，这个数据即可以保存在Hive元数据库中，也可以保存在一个配置文件中。为了方便起见，这里使用配置文件来保存。

首先需要编写用户权限验证的类：

package com.lxw1234.hive.auth;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import javax.security.sasl.AuthenticationException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.hive.conf.HiveConf;

import org.apache.hive.service.auth.PasswdAuthenticationProvider;

public class CustomHiveServer2Auth implements PasswdAuthenticationProvider  {

    @Override

    public void Authenticate(String username, String password)

            throws AuthenticationException {

        boolean ok = false;

        String passMd5 = new MD5().md5(password);

        HiveConf hiveConf = new HiveConf();

        Configuration conf = new Configuration(hiveConf);

        String filePath = conf.get("hive.server2.custom.authentication.file");

        System.out.println("hive.server2.custom.authentication.file [" + filePath + "] ..");

        File file = new File(filePath);

        BufferedReader reader = null;

        try {

            reader = new BufferedReader(new FileReader(file));

            String tempString = null;

            while ((tempString = reader.readLine()) != null) {

                String[] datas = tempString.split(",", -1);

                if(datas.length != 2) continue;

                //ok

                if(datas[0].equals(username) && datas[1].equals(passMd5)) {

                    ok = true;

                    break;

                }

            }

            reader.close();

        } catch (Exception e) {

            e.printStackTrace();

            throw new AuthenticationException("read auth config file error, [" + filePath + "] ..", e);

        } finally {

            if (reader != null) {

                try {

                    reader.close();

                } catch (IOException e1) {}

            }

        }

        if(ok) {

            System.out.println("user [" + username + "] auth check ok .. ");

        } else {

            System.out.println("user [" + username + "] auth check fail .. ");

            throw new AuthenticationException("user [" + username + "] auth check fail .. ");

        }

    }

    //MD5加密

    class MD5 {

        private MessageDigest digest;

        private char hexDigits[] = {'0','1','2','3','4','5','6','7','8','9','a','b','c','d','e','f'};

        public MD5() {

            try {

              digest = MessageDigest.getInstance("MD5");

            } catch (NoSuchAlgorithmException e) {

              throw new RuntimeException(e);

            }

        }

         public String md5(String str) {

            byte[] btInput = str.getBytes();

            digest.reset();

            digest.update(btInput);

            byte[] md = digest.digest();

            // 把密文转换成十六进制的字符串形式

            int j = md.length;

            char strChar[] = new char[j \* 2];

            int k = 0;

            for (int i = 0; i < j; i++) {

                byte byte0 = md[i];

                strChar[k++] = hexDigits[byte0 >>> 4 & 0xf];

                strChar[k++] = hexDigits[byte0 & 0xf];

            }

            return new String(strChar);

        }

    }

}

将上面的程序打包成HiveServer2Auth.jar，放到$HIVE\_HOME/lib下，  
在hive-site.xml中设置以下参数：  
<property>  
<name>hive.server2.authentication</name>  
<value>CUSTOM</value>  
</property>  
<property>  
<name>hive.server2.custom.authentication.class</name>  
<value>com.lxw1234.hive.auth.CustomHiveServer2Auth</value>  
</property>  
<property>  
<name>hive.server2.custom.authentication.file</name>  
<value>/usr/local/apache-hive-0.13.1-bin/conf/hive.server2.users.conf</value>  
</property>

在$HIVE\_HOME/conf下新建文件hive.server2.users.conf，里面写入内容：  
[root@dev conf]# cat hive.server2.users.conf  
lxw1234,48d9a656690e1b1bf5ddee4c12d1bbd7  
user,5f4dcc3b5aa765d61d8327deb882cf99

其中，48d9a656690e1b1bf5ddee4c12d1bbd7为lxw1234\_password的md5加密，  
5f4dcc3b5aa765d61d8327deb882cf99为password的md5加密。

Idea debug hive --debug[:port=18000]

Hive客户端查看日志：

Hive --hiveconf hive.root.logger=DEBUG,console

**Hive执行流程：ql.Driver类**

**Parse -- semantic analyses -- execute**

Driver.run()

=>HiveDriverRunHook.preDriverRun()( hive.exec.driver.run.hooks )

=>Driver.compile()

=>HiveSemanticAnalyzerHook.preAnalyze()( hive.semantic.analyzer.hook )

=>SemanticAnalyze(QueryBlock, LogicalPlan, PhyPlan, TaskTree)

=>HiveSemanticAnalyzerHook.postAnalyze()( hive.semantic.analyzer.hook )

=>QueryString redactor( hive.exec.query.redactor.hooks )

=>QueryPlan Generation

=>Authorization

=>Driver.execute()

=>ExecuteWithHookContext.run() || PreExecute.run() ( hive.exec.pre.hooks )

=>TaskRunner

=>if failed, ExecuteWithHookContext.run()( hive.exec.failure.hooks )

=>ExecuteWithHookContext.run() || PostExecute.run() ( hive.exec.post.hooks )

=>HiveDriverRunHook.postDriverRun()( hive.exec.driver.run.hooks )