



Hive 使用手册

内部技术文档 2015 年夏

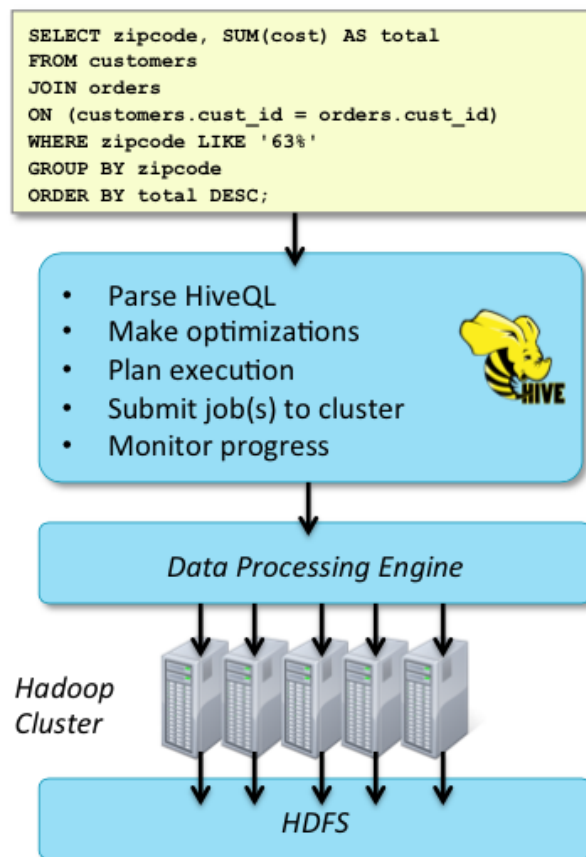
版本修改历史			
版本	作者	描述	日期
1.0	丁晔磊	第一版	2015.06.09

目录

Hive 基本概念.....	4
Hadoop 基本配置	4
Zookeeper 基本配置.....	5
HDFS 基本配置.....	5
Yarn 基本配置.....	6
Yarn 调度算法.....	7
Yarn 动态资源池.....	8
Sentry 授权.....	11
Sentry 授权模型.....	12
Sentry 授权实例.....	13
Kerberos 认证	14
LDAP 认证	16
Hive/Impala 互操作	17

Hive 基本概念

Apache Hive 在 MapReduce 上提供了一个 SQL 引擎层，是 Facebook 开发并开源的一个 Apache 项目。Apache Hive 支持 HiveQL 语言，是 SQL-92 的一个子集。Hive 只是提供了一个 SQL 的解析，具体的执行依赖于底层的执行引擎，比如 MapReduce。Hive 将 HiveQL 查询转换为一系列的 MapReduce 作业，提交到集群中运行，如下图所示：



Hadoop 基本配置

Hive 依赖的组件包括 Yarn、HDFS 与 Zookeeper。为了 Hive 的正常工作，Cloudera 对相关组件有一些推荐配置，主要包括角色并发性、任务资源配置等。

Zookeeper 基本配置

配置并发连接数，默认值是 60，在高并发的情况下导致访问集群连接失败(connection refused)的错误。建议调整到 2000。

 **ZooKeeper** [Status](#) [Instances](#) [Configuration](#) [Commands](#) [Audits](#) [Charts Library](#)

Configuration

<input type="text" value="maxClient"/> <input type="button" value="x"/>		
Category	Property	Value
Server Default Group	Maximum Client Connections maxClientCnxns	<input type="text" value="2000"/> Reset to the default value: 60

HDFS 基本配置

调整 NameNode 内存的使用量 (同时相应调整 Secondary NameNode 内存的使用量)，默认值是 1GB。建议调整到至少 4GB。

 **HDFS** [Status](#) [Instances](#) [Configuration](#) [Commands](#) [Audits](#) [File Browser](#) [Charts](#)

Configuration

<input type="text" value="Search"/> <input type="button" value="x"/> Role Groups History and I		
Category	Property	Value
▶ Service-Wide	Java Heap Size of Namenode in Bytes	<input type="text" value="4"/> GiB Reset to the default value: 1 GiB <small>Java Heap Size of Namenode in Bytes is at least 1GB for every million HDFS blocks. Suggested minimum value: 1073741824</small>
▶ Balancer Default Group		
▶ DataNode Default Group		
▶ Failover Controller Default Group		
▶ Gateway Default Group		

调整 DataNode 的 Handler 的数量，默认值是 3。建议调整到 32 或者 64。

 **HDFS** [Status](#) [Instances](#) [Configuration](#) [Commands](#) [Audits](#) [File Browser](#) [Charts](#)

Configuration

<input type="text" value="dfs.datanode.handler.count"/> <input type="button" value="x"/> Role Groups		
Category	Property	Value
DataNode Default Group / Performance	Handler Count dfs.datanode.handler.count	<input type="text" value="64"/> Reset to the default value: 3

开启 HDFS ACLs (Access Control Lists)，默认是关闭的。建议开启，可以更加灵活地管理 HDFS 文件系统的访问权限。

Configuration

<input type="text" value="access control lists"/> ✕		
Category	Property	Value
Service-Wide / Security	Enable Access Control Lists dfs.namenode.acls.enabled	<input checked="" type="checkbox"/> Reset to the default value: false ↶

Yarn 基本配置

调整 Map Container 的内存使用量，默认值是 1GB。推荐使用 2GB 作为起始点，根据应用程序的运行状况进行调整。

Configuration

<input type="text" value="map.memory"/> ✕ Role Groups History		
Category	Property	Value
Gateway Default Group / Resource Management	Map Task Memory mapreduce.map.memory.mb	<input type="text" value="2"/> GiB Reset to the default value: 1 GiB ↶

调整 Reduce Container 的内存使用量，默认值是 1GB。推荐使用 4GB 作为起始点，根据应用程序的运行状况进行调整。

Configuration

<input type="text" value="reduce.memory"/> ✕ Role Groups History a		
Category	Property	Value
Gateway Default Group / Resource Management	Reduce Task Memory mapreduce.reduce.memory.mb	<input type="text" value="4"/> GiB Reset to the default value: 1 GiB ↶

注意对于以上两个参数的调整，需要调整相应的 java.opts 参数以设置 JVM 的内存堆栈大小。

调整 Map/Reduce 任务(排序用)内存缓冲区大小，默认值是 256MB。推荐使用 512MB 作为起始点，根据应用程序的运行状况进行调整。

YARN (MR2 Included)

Status

Instances

Configuration

Commands

Audits

Application

Configuration

Q

sort.mb

✕

Role Groups

History

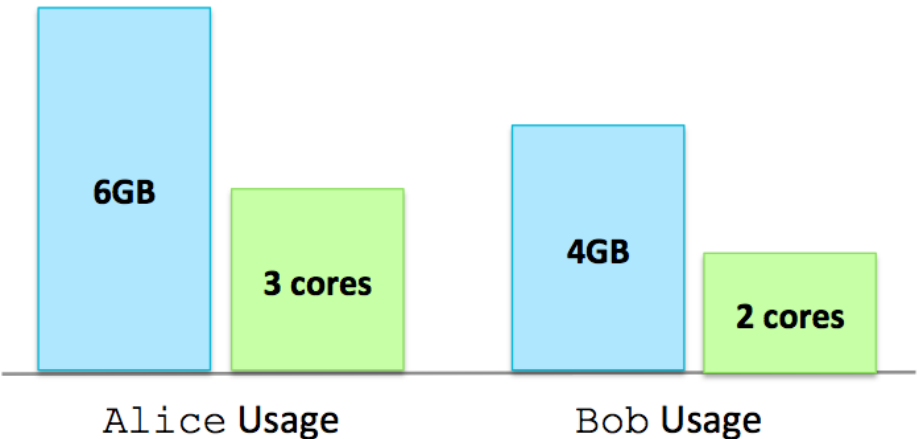
Category	Property	Value	Description
Gateway Default Group	I/O Sort Memory Buffer (MiB) mapreduce.task.io.sort.mb	<div>512</div> <div>MiB</div> <div>Reset to the default value: 256 MiB</div>	The total i files. Note (meaning usable he

Yarn 调度算法

Yarn 的(Fair Scheduler)调度算法主要有 3 类：DRF (Dominant Resource Allocation)、FAIR、FIFO (First In First Out)。默认使用 DRF，利用 CPU 以及 Memory 实现资源的公平调度。FAIR 算法与 DRF 相似，资源是在多任务之间共享的，区别在于 FAIR 算法仅仅根据 Memory 实现资源的调度。FIFO 算法，顾名思义，提交到某一个资源池的多个任务根据提交的时间顺序依次运行，后提交的任务在先提交的任务执行完毕前不会被调度(即 Resource Manager 不会为后提交的任务分配运行 Application Master 的 Container)。

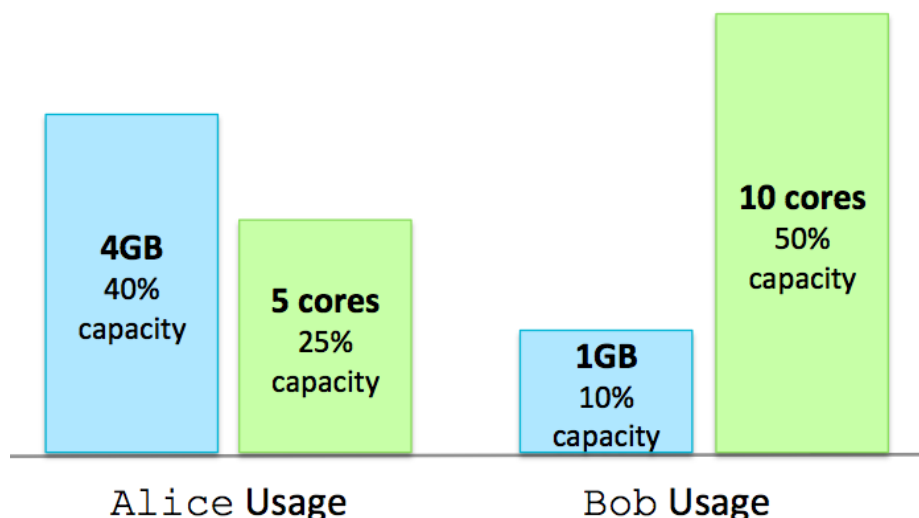
DRF 调度算法举例，假设两个用户 Alice 与 Bob 分别提交任务：

情况一：Alice 获得 6GB 内存、3 个 CPU 核；Bob 获得 4GB 内存、2 个 CPU 核



该情况下，无论是 CPU 还是 Memory，Alice 都具有绝对优势，因此 Resource Manager 在分配下一个 Container 时会先满足 Bob 的资源请求。

情况二： 集群总共有 10GB 内存、20 个 CPU 核。Alice 获得 4GB 内存、5 个 CPU 核；Bob 获得 1GB 内存、10 个 CPU 核



该情况下，Alice 拥有更多的内存(40% > 10%) 而 Bob 拥有更多的 CPU 核 (50% > 25%)。但是相比而言 50% > 40%，因此 Resource Manager 在分配下一个 Container 时会先满足 Alice 的资源请求。

Yarn 动态资源池

任何 Yarn 应用程序的运行都是在动态资源池中进行的，至于具体使用哪个资源池，用户可以手工指定(比如 MapReduce 应用程序，通过参数 `mapreduce.job.queueName` 确定)，也可以通过 Resource Manager 的“Placement Rules”自动分配。

默认情况下，“Placement Rules”使用 `root.USER_NAME`。例如用户 alex 提交一个 MapReduce 作业，在没有显示使用 `mapreduce.job.queueName` 的条件下，Resource Manager 会自动将该作业放置到资源池 `root.alex` 中。若 `root.alex` 资源池尚未创建，Resource Manager 会自动(动态)创建该资源池。重启 Yarn 服务后，所有动态创建的资源池会被自动删除。

使用动态创建的资源池并不能非常严格地控制资源的使用。推荐使用“Placement Rules”的高级配置，预分配后续需要的资源池。如下图所示，Resource Manager 资源池选择过程为：

判断用户 `alex:prod` 是否显示设置参数 `mapreduce.job.queueName`

- ➔ 是并且该资源池预定义，在指定资源池中运行
- ➔ 否，判断资源池 `root.prod` 是否预定义
 - ➔ 是，在 `root.prod` 资源池中运行

➔ 否，在 default 资源池中运行

Dynamic Resource Pools Status Configuration

Resource Pools Scheduling Rules Placement Rules User Limits Other Settings

Applications can run in a pool based on the user, the group of the submitting user, as well as **specific** pools and the default pool. Configure how an application will determine in which pool it will run.

☐ Basic

☒ Advanced

Specify the order in which rules are evaluated to determine in which pool an application will run.

If a rule is always satisfied, subsequent rules are not evaluated and appear disabled. If a rule has a condition that is not

specified pool only if the pool exists. ▼

-

+

root.<primaryGroupName> pool only if the pool exists. ▼

-

+

default pool; create the pool if it doesn't exist. ▼

-

This rule is always

用户允许为每个资源池配置不同的调度算法、资源的约束(资源池权重、CPU 核数、内存数量、最大运行应用程序个数)、提交访问控制(Submission Access Control)、管理访问控制(Administration Access Control)。子资源池会自动继承父资源池的提交访问控制以及管理访问控制。例如，如果允许用户 alex 向 root 资源池提交应用程序，那么该用户可以向任意 root 的子资源池，比如 root.dev、root.prod，提交应用程序。

Edit Resource Pool: root

General YARN Submission Access Control Administration Access Control

This feature is relevant only if **Enable ResourceManager ACLs** is set to **true** and **Admin ACL** is NOT set to * (See Other Settings).

Fair Scheduler Access Control Lists control who can submit applications to pools. For subpools, users who have permission to submit a parent pool automatically inherit the same ability for the child.

☐ Allow anyone to submit to this pool

☒ Allow these users and groups to submit to this pool

Users

alex

Groups

Comma separated list of groups. Space characters are not allowed.

OK

Cancel



Cloudera, Inc. 1001 Page Mill Road, Palo Alto, CA 94304

1-888-789-1488 or 1-650-362-0488

cloudera.com

©2015 Cloudera, Inc. All rights reserved. Cloudera and the Cloudera logo are trademarks or registered trademarks of Cloudera Inc. in the USA and other countries. All other trademarks are the property of their respective companies. Information is subject to change without notice.

资源池的属性是允许动态修改的，保存在 `fair-scheduler.xml` 文件中。Resource Manager 每 10 秒会自动读取该文件，刷新资源池的属性。如果需要人为编写 `fair-scheduler.xml`，使用 Yarn 配置参数 “Fair Scheduler XML Advanced Configuration Snippet (Safety Valve)”：

[YARN \(MR2 Included\)](#) [Status](#) [Instances](#) [Configuration](#) [Commands](#) [Audits](#) [Applications](#) [Charts Library](#)

Configuration

Role Groups

Category	Property	Value	Desc
ResourceManager Default Group	Fair Scheduler XML Advanced Configuration Snippet (Safety Valve)	Default value is empty. Click to edit.	An XML snippet that is used to configure the Fair Scheduler. The snippet is added to the <code>fair-scheduler.xml</code> file.

设置资源池提交访问控制、管理访问控制后，用户通过 Yarn 内嵌管理页面查看应用程序时可能会遇到如下问题：



Cluster

[About](#)
[Nodes](#)
[Applications](#)

You (User dr.who) are not authorized to view application application_1433384142873_0003

加入参数 `hadoop.http.staticuser.user`：

[YARN \(MR2 Included\)](#) [Status](#) [Instances](#) [Configuration](#) [Commands](#) [Audits](#) [Applications](#) [Charts Library](#)

Configuration

Service-Wide / Advanced

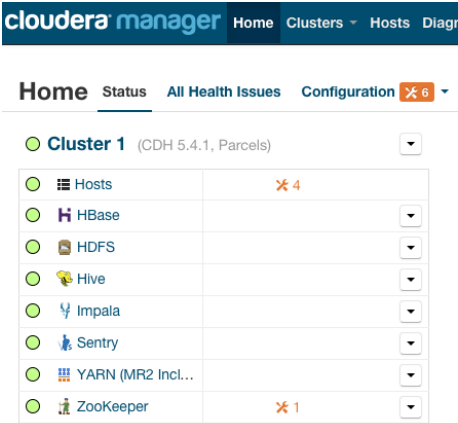
YARN Service Advanced Configuration Snippet (Safety Valve) for core-site.xml

```
<property>
  <name>hadoop.http.staticuser.user</name>
  <value>yarn</value>
</property>
```

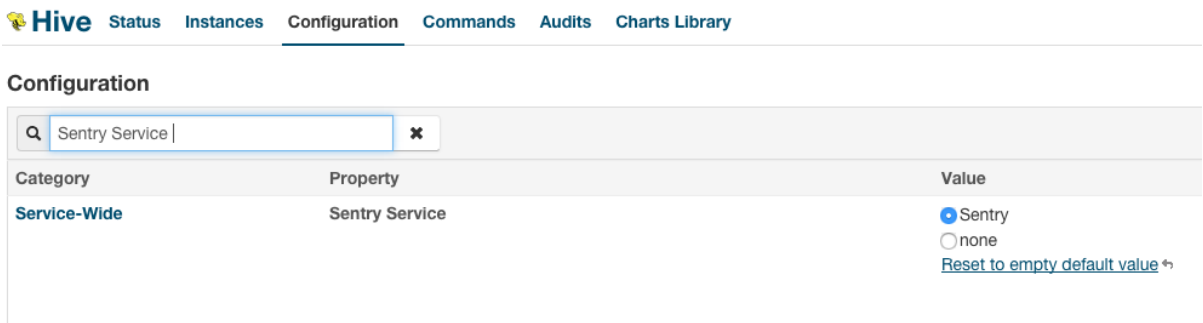
[Reset to the default value: ...](#)

Sentry 授权

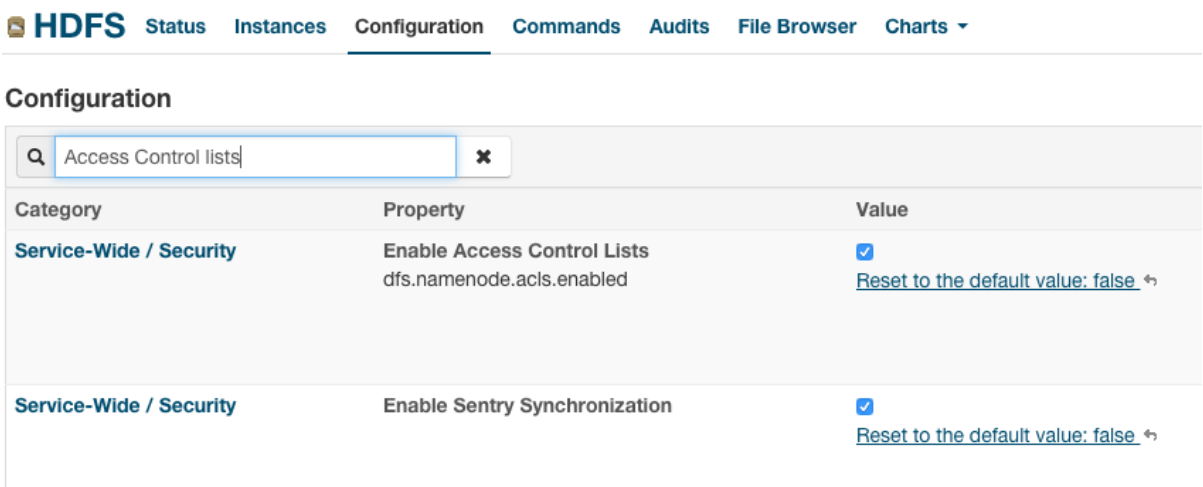
虽然推荐在 Kerberos 模式下开启 Sentry 授权，但是用户也允许在普通模式下使用 Sentry。安装 Sentry 可以采用常规“Add a Service”的流程。安装完毕后可以在 Cloudera Manager 主页看到新添加的 Sentry 服务。




配置 Hive 采用 Sentry 授权：



配置 HDFS 开启 ACLs (Access Control Lists) 与 SentryHDFS 权限同步：



关闭 Hive Impersonation 功能:

 **Hive** [Status](#) [Instances](#) [Configuration](#) [Commands](#) [Audits](#) [Charts Library](#)

Configuration


✕

Category	Property	Value
HiveServer2 Default Group	HiveServer2 Enable Impersonation hive.server2.enable.impersonation, hive.server2.enable.doAs	<input type="checkbox"/> Reset to the default value: true ↩

为了在非 Kerberos 环境下使用 Sentry 授权, 需要设置参数 `sentry.hive.testing.mode` 为 `true` (加入 Hive Service Advanced Configuration Snippet for sentry-site.xml 中)。

Sentry 授权模型

Sentry 的授权是基于对象进行的, 最常用的包括服务器(Server)、数据库(Database)、表(Table)、视图(View)。后面三者比较好理解, 第一个 Server 指代整个 Hive Service, 默认值是 `server1`:

 **Hive** [Status](#) [Instances](#) [Configuration](#) [Commands](#) [Audits](#) [Charts Library](#)

Configuration

✕

Category	Property	Value
Service-Wide / Advanced	Server Name for Sentry Authorization hive.sentry.server	server1 default value

目前最新版本 CDH 所包含的 Hive 并不支持基于列的授权。因此如果需要通过列级别的授权, 可以使用视图(view)。Sentry 的权限有三种: INSERT、SELECT、ALL:

权限	对象
INSERT	DB、TABLE
SELECT	DB、TABLE
ALL	SERVER、TABLE、DB、URI

一般 Sentry 授权流程: 超级用户将“权限”授权给“角色”→超级用户将“角色”授权给“用户组”。用户组中的所有用户享用赋予的权限, 这里提到的用户组可以是用户的 **Primary Group**, 也可以是 **Secondary Groups**。另外, 还需要注意的是, Sentry 中“角色”只能授权给“用户组”, 并不能直接授权给特定“用户”。

Sentry 授权实例

案例：某公司数据中心部门在 Hive 中部署数据库 sjzx，现在希望对两类用户——管理员与开发人员——实现权限的控制。管理员对数据库 sjzx 享有所有权限；开发人员对数据库 sjzx 仅享有只读权限。

步骤 1：在集群中新建用户/用户组

```
groupadd -g 3001 sjzx
useradd -u 3001 -g sjxsjzx
groupadd -g 3002 sjzx_dev
useradd -u 3002 -g sjzx_devsjzx_dev
```

步骤 2：HDFS 超级用户 hdfs 创建用户工作目录

```
hdfsdfs -mkdir /user/sjzx
hdfsdfs -chownsjzx:sjzx /user/sjzx
hdfsdfs -mkdir /user/sjzx_dev
hdfsdfs -chownsjzx_dev:sjzx_dev /user/sjzx_dev
```

步骤 3：HDFS 超级用户 hdfs 对“Hive 内部表数据存储目录”设置正确的权限

```
hdfsdfs -chmod -R 771 /user/hive/warehouse
hdfsdfs -chown -R hive:hive /user/hive/warehouse
```

步骤 4：Hive 超级用户 hive 创建数据库并授权

```
beeline>create role admin;
beeline>grant all on server to role admin;
beeline> grant role admin to group hive;
beeline> create database sjzx;
beeline> create role sjzx_dev;
beeline> grant role sjzx_dev to group sjzx_dev;
beeline> grant select on database sjzx to role sjzx_dev;
beeline> create role sjzx;
beeline> grant role sjzx to group sjzx;
beeline> grant all on database sjzx to role sjzx;
```

步骤 5：数据中心管理员 sjzx 验证

```
#权限验证
beeline>use sjzx;

beeline>create table t1 (name string, company string) row format delimited fields terminated by ',';
beeline>show tables;
Query: show tables
```

```

+-----+--+
| tab_name |
+-----+--+
| t1      |
+-----+--+

# 编写测试数据加入内部表 sjzx.t1，例如 “alex,cloudera”
#数据查询验证
beeline>select * from t1;
+-----+-----+--+
| t1.name | t1.company |
+-----+-----+--+
| alex   | cloudera  |
+-----+-----+--+

```

步骤 6: 数据中心开发员 sjzx_dev 验证

```

beeline>use sjzx;

beeline>show tables;
+-----+--+
| tab_name |
+-----+--+
| t1      |
+-----+--+

beeline>select * from t1;
+-----+-----+--+
| t1.name | t1.company |
+-----+-----+--+
| alex   | cloudera  |
+-----+-----+--+

beeline>create table t2 (name string, company string) row format delimited fields terminated by ',';
Error: Error while compiling statement: FAILED: SemanticException No valid privileges
Required privileges for this query: Server=server1->Db=sjzx->action=*; (state=42000,code=40000)

```

Sentry 授权是通过 HiverServer2 实现的，因此为了使用授权功能就必须使用 beeline 客户端 (Hive CLI 并不经过 HiveServer2，因此会绕过授权管理部分，通过步骤 3 在 HDFS 文件系统级别屏蔽 Hive CLI)。

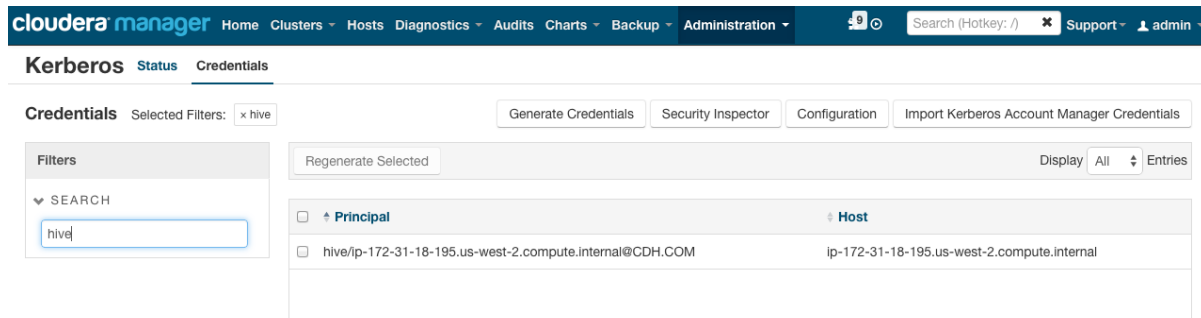
Kerberos 认证

Hadoop 安全包括认证、授权和审计。其中认证是安全的第一步，是授权和审计的基础。Hadoop 默认情况下采用 Linux 的用户、用户组验证 HDFS 文件权限，Yarn 动态资源池的权限等。如果要

实现强认证，推荐使用 Kerberos。当前常用的两种实现是 MIT KDC 以及 Active Directory。前者是 Kerberos 开源实现版本；后者不仅实现了 Kerberos 协议，同时也实现了 LDAP 协议。

Hive 的认证其实分为两个阶段：第一阶段，客户端到 HiveServer2 的认证，允许使用 Kerberos、LDAP 实现认证；第二阶段，HiveServer2 到 HDFS、Yarn 等 Hadoop 后台服务的认证，使用 Kerberos 实现认证。当集群开启 Kerberos 认证后，两个阶段都默认使用 Kerberos 进行认证，其中第一阶段使用用户自己的 Kerberos 账户，第二阶段使用运行 HiveServer2 的 Kerberos 账户。

在 Cloudera Manager 页面，管理员可以看到 HiveServer2 的 Kerberos 账户：



运行 beeline 前，普通用户需要使用其 Kerberos 账户进行登录，例如：

```
kinitalex
Password for alex@CDH.COM:

klist
Ticket cache: FILE:/tmp/krb5cc_500
Default principal: alex@CDH.COM

Valid starting Expires Service principal
06/04/15 13:20:34 06/04/15 23:20:40 krbtgt/CDH.COM@CDH.COM
renew until 06/11/15 13:20:34
```

否则在使用 beeline 连接 HiveServer2 的过程中，会出现错误：

```
15/06/04 13:32:11 [main]: ERROR transport.TSaslTransport: SASL negotiation failure
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: No valid
credentials provided (Mechanism level: Failed to find any Kerberos tgt)]
```

在启动 Kerberos 后，beeline 连接 HiveServer2 时，在指定的 JDBCURL 路径末尾添加 HiveServer2 的 principal 名称：



```
beeline
Beeline version 1.1.0-cdh5.4.1 by Apache Hive
beeline>!connect jdbc:hive2://ip-172-31-18-195.us-west-
2.compute.internal:10000/default;principal=hive/ip-172-31-18-195.us-west-
2.compute.internal@CDH.COM
scan complete in 4ms
```

```
Connecting to jdbc:hive2://ip-172-31-18-195.us-west-2.compute.internal:10000/default;principal=hive/ip-172-31-18-195.us-west-2.compute.internal@CDH.COM
Enter username for jdbc:hive2://ip-172-31-18-195.us-west-2.compute.internal:10000/default;principal=hive/ip-172-31-18-195.us-west-2.compute.internal@CDH.COM:
Enter password for jdbc:hive2://ip-172-31-18-195.us-west-2.compute.internal:10000/default;principal=hive/ip-172-31-18-195.us-west-2.compute.internal@CDH.COM:
Connected to: Apache Hive (version 1.1.0-cdh5.4.1)
Driver: Hive JDBC (version 1.1.0-cdh5.4.1)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://ip-172-31-18-195.us-west-2.co>
```

beeline 连接过程中会提示用户输入用户名(username)、密码(password)，直接回车即可 (实际 HiveServer2 认证时只会判断启动 beeline 用户的 Kerberos token)。

LDAP 认证

在 Kerberos 环境下，客户端到 HiveServer2 的认证还可以通过 LDAP 的方式进行。默认情况下，Hive 中 LDAP 的配置都是“Service-Wide”的。


 Status Instances Configuration Commands Audits Charts Library		
Configuration		
<input type="text" value="LDAP"/> 		
Category	Property	Value
Service-Wide / Security	Enable LDAP Authentication	<input type="checkbox"/> default value
Service-Wide / Security	LDAP URI hive.server2.authentication.ldap.url	Default value is empty. Click to edit.
Service-Wide / Security	LDAP Domain hive.server2.authentication.ldap.Domain	Default value is empty. Click to edit.
Service-Wide / Security	LDAP BaseDN hive.server2.authentication.ldap.baseDN	Default value is empty. Click to edit.

从参数的命名可以发现，LDAP 相关的设置都是基于 HiveServer2 的，即允许用户启动多个 HiveServer2 实例 (instance)，不同实例采用不同的认证方式，提高 Hive 的使用人群范围。比如，在 Kerberos 环境中，用户必须拥有 Kerberos 凭证才可以使用 Hive 服务，但是如果用户拥有了 Kerberos 凭证，那么该用户就被允许使用 Hadoop 中所有的服务了，如果只希望将 Hive 服务暴露给外部用户使用，可以采用 LDAP 认证的方式。

LDAP 协议的两种常用实现是 OpenLDAP 和 Active Directory。因此针对不同的实现，需要配置不同的参数：

参数	适用实现方式
hive.server2.authentication	OpenLDAP、Active Directory
hive.server2.authentication.ldap.url	OpenLDAP、Active Directory
hive.server2.authentication.ldap.Domain	Active Directory
hive.server2.authentication.ldap.baseDN	OpenLDAP

以 Windows Server 2008 R2 Active Directory 为例，需要配置上述参数列表中的前三项。修改 HiveServer2 配置，覆盖 Kerberos 认证方式：

 [Status](#) [Instances](#) [Configuration](#) [Commands](#) [Audits](#) [Charts Library](#)

Configuration

LDAP HiveServer2 Advanced Configuration ✕

Category

Property

Value

HiveServer2 LDAP Group / Advanced

HiveServer2 Advanced Configuration Snippet (Safety Valve) for hive-site.xml

```
<property>
  <name>hive.server2.authentication</name>
  <value>LDAP</value>
</property>
<property>
  <name>hive.server2.authentication.ldap.url</name>
  <value>ldap://ip-172-31-18-43.us-west-2.compute.internal</value>
</property>
<property>
  <name>hive.server2.authentication.ldap.Domain</name>
  <value>cdh.com</value>
</property>
```

[Reset to the default value: ...](#)

重启 HiveServer2 实例后，就可以使用 beeline 进行访问了(无需 Kerberos Token)，由于采用 LDAP 认证，连接 HiveServer2 时直接使用普通的 JDBC URL 即可。根据 LDAP 中配置的用户名/密码登录。

```
beeline
beeline>!connect jdbc:hive2://ip-172-31-18-194.us-west-2.compute.internal:10000/default
Connecting to jdbc:hive2://ip-172-31-18-194.us-west-2.compute.internal:10000/default
Enter username for jdbc:hive2://ip-172-31-18-194.us-west-2.compute.internal:10000/default:
sjzx_dev
Enter password for jdbc:hive2://ip-172-31-18-194.us-west-2.compute.internal:10000/default:
*****
Connected to: Apache Hive (version 1.1.0-cdh5.4.1)
```

Hive/Impala 互操作

Hive 与 Impala 本身是共享 MetaStore 的，即 Hive(或者 Impala)对表的操作都会反映到 Impala(或者 Hive)中。Hive 与 Impala 的授权都是通过 Sentry 实现的，进而提供了一个统一的 SQL 授权访问层(推荐 Hive/Impala 设置统一的认证方式)。

从性能角度出发，推荐使用 Parquet 文件存储格式，这种列式存储方式可以有效降低 table scan 过程中读取数据的数量。实际操作中，使用 Impala 生成 Parquet 数据时，每个输出文件的

大小基本一致 (~256MB)。但是如果利用 Hive 生成 Parquet 数据，例如 `insert overwrite table t1_parquet select * from t1_text`，每个 Parquet 文件会出现过小、大小不均匀等问题。需要对一些参数进行调整以解决这一系列问题：

参数	描述
<code>mapreduce.input.fileinputformat.split.maxsize</code>	Mapper 任务处理的 InputSplit 最大值
<code>mapreduce.input.fileinputformat.split.minsize</code>	Mapper 任务处理的 InputSplit 最小值
<code>dfs.blocksize</code>	底层 HDFS 存储时 Block 的大小
<code>parquet.block.size</code>	生成 Parquet 时使用内存缓冲区大小
<code>parquet.compression</code>	压缩算法

运行 hive 生成 Parquet 示例：

```
setdfs.blocksize=268435456;
setparquet.compression=snappy;

#不同文件的压缩率不同，需要调整以下3个参数以保证最终生成的文件小于一个HDFS block
setmapred.min.split.size=805306368;
setmapred.max.split.size=805306368;
setparquet.block.size=536870912;

drop table if exists hive_demo;
create table hive_demo
stored as parquet
as
select * from t1;
```

如果生成的 Impala 小文件超过一个 HDFS Block (执行时需要通过网络读取超出的部分)，那么在运行 Impala 查询时会出现警告：

```
WARNINGS: Parquet files should not be split into multiple hdfs-blocks.
file=hdfs://../warehouse/hive_demo/000000_0 (1 of 5 similar)
```