

英特尔® Apache Hadoop* 软件发行版安全指南

版本 2.3

2013 年 3 月



免责声明和法律信息

本文件中包含关于英特尔产品的信息。本文件不构成对任何知识产权的授权，包括明示的、暗示的，也无论是基于禁止反言的原则或其他。除英特尔产品销售的条款和条件规定的责任外，英特尔不承担任何其他责任。英特尔在此作出免责声明：本文件不构成英特尔关于其产品的使用和 / 或销售的任何明示或暗示的保证，包括不就其产品的 (i) 对某一特定用途的适用性、(ii) 适销性以及 (iii) 对任何专利、版权或其他知识产权的侵害的承担任何责任或作出任何担保。

除非经过英特尔的书面同意认可，英特尔的产品无意被设计用于或被用于以下应用：即在这样的应用中可因英特尔产品的故障而导致人身伤亡。

英特尔有权随时更改产品的规格和描述而无需发出通知。设计者不应信赖任何英特尔产品所不具有的特性，设计者亦不应信赖任何标有“保留权利”或“未定义”说明或特性描述。对此，英特尔保留将来对其进行定义的权利，同时，英特尔不应为其日后更改该等说明或特性描述而产生的冲突和不相容承担任何责任。此处提供的信息可随时改变而无需通知。请勿根据本文件提供的信息完成一项产品设计。

本文件所描述的产品可能包含使其与宣称的规格不符的设计缺陷或失误。这些缺陷或失误已收录于勘误表中，可索取获得。

在发出订单之前，请联系当地的英特尔营业部或分销商以获取最新的产品规格。

索取本文件中或英特尔的其他材料中提的、包含订单号的文件的复印件，可拨打 1-800-548-4725，或登陆 <http://www.intel.com/design/literature.htm>。

英特尔处理器标号不是性能的指标。处理器标号仅用于区分同属一个系列的处理器的特性，而不能够用于区分不同系列的处理器。**详情请登陆：**
http://www.intel.com/products/processor_number

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit [Intel Performance Benchmark Limitations](#).

结果基于模拟测算得出，仅作参考之用。结果通过系统模拟器或模型测算得出。任何系统硬件、软件的设计或配置的不同均可能影响实际性能。

Intel, 英特尔® Apache Hadoop® 软件发行版, Intel® Distribution, Intel® Manager for Apache Hadoop® software, Intel® Manager 是英特尔在美国和 / 或其他国家的商标。

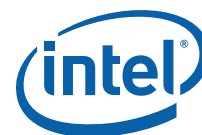
* 其他的名称和品牌可能是其他所有者的资产。

英特尔公司 2013 年版权所有。所有权保留。



文档修订记录

日期	修订	描述
2012 年 12 月	001	英特尔® Apache Hadoop* 软件发行版 v2.2 第一版
2013 年 2 月	002	英特尔® Apache Hadoop* 软件发行版 v2.3 文档更新
2013 年 2 月	003	英特尔® Apache Hadoop* 软件发行版 v2.3 文档更新
2013 年 2 月	004	英特尔® Apache Hadoop* 软件发行版 v2.3 文档更新
2013 年 3 月	005	英特尔® Apache Hadoop* 软件发行版 v2.3 文档更新

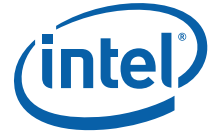


目录

1.0	为 Apache Hadoop* 服务使用 Kerberos* 认证	6
1.1	Kerberos 在 Apache Hadoop* 集群是如何应用的?	6
1.2	先决条件和推荐配置	6
1.3	为 Apache Hadoop* 配置 Kerberos	7
1.3.1	定义 Kerberos 领域	7
1.3.2	配置 Kerberos 化的 SSL	8
1.3.3	安装和配置 Kerberos 客户端	8
1.4	在 Intel® Manager 中更改安全模式	9
1.5	创建密钥表	10
1.5.1	生成密钥表示例	11
1.6	在 Apache Hadoop* 集群中部署密钥表	13
1.7	在安全模式中测试 Apache Hadoop* 功能	13
1.8	解决 Kerberos 配置问题	14
1.8.1	客户端访问失败并显示错误信息 SASL authentication failed	14
1.8.2	客户端访问失败并显示错误信息 GSS initiate Failed	14
2.0	为 Apache Hadoop* 服务设置基于角色的验证	15
2.1	功能和作用	15
2.2	访问控制列表是如何工作的?	15
2.3	什么是角色?	15
2.4	先决条件	19
2.5	设置和 Active Directory 服务器的通讯	19
2.6	在集群中部署基于角色的验证变化	21
2.7	控制 LDAP 组的缓存	22
2.8	基于角色验证的用户场景	22
2.8.1	定义角色	23
2.8.2	分配角色给用户	23
2.8.3	设置一个 Active Directory 和 Unix 账户	23
2.8.4	启用基于角色的验证 Intel® Manager	24
2.8.5	为基于角色的验证配置 MapReduce 公平调度器	24
2.8.6	在 Intel® Manager 中设置 hadoop-superusers 角色	25
2.8.7	在 Intel® Manager 中设置 access_mapredqueue_development 角色	29
2.8.8	在 Intel® Manager 中设置 access_hbasetable_phonerecords 角色	31
2.8.9	在 Intel® Manager 中设置 access_hivetable_warehouse 角色	32
3.0	使用应用层加密实现数据保密	35
3.1	加密的工作原理	35
3.2	支持的加密方式	35
3.2.1	对称加密	35
3.2.2	非对称加密	35
3.3	密钥是如何管理的?	36
3.4	需要加密的 API 和 JavaDoc 存在哪里?	36
3.5	编辑执行加密的 Java 程序	36
3.6	先决条件	36
3.6.1	总体要求	36
3.6.2	非对称加密的要求	36
3.6.3	对称加密的要求	37
3.6.4	KeyStore 密码文件的要求	37
3.7	创建 Java KeyStore	38
3.7.1	创建对称加密的 Java KeyStore	38
3.7.2	创建非对称加密的 Java KeyStore 和 TrustStore	39
3.8	创建加密和解密数据的 HDFS 应用程序	42



3.9	创建加密和解密数据的 MapReduce 应用程序	44
-----	---------------------------------	----



1.0 为 Apache Hadoop* 服务使用 Kerberos* 认证

Kerberos* 是一种网络认证协议，其设计目标是通过对称密钥体制为客户机 / 服务器应用程序提供强大的认证服务。这一协议要求共同认证，也就是说，在客户端允许使用服务器端资源之前，客户端和服务器端必须相互认证对方身份。Kerberos 认证的目的在于让非加密网络的应用程序在通讯时，通过加密的方式向对方认证它们的身份。

以下章节解释了如何为英特尔® Apache Hadoop* 软件发行版设置 Kerberos 认证

1.1 Kerberos 在 Apache Hadoop* 集群是如何应用的？

在 Apache Hadoop* 集群中，Kerberos 认证通过以下方式被应用：

- 可信任服务 — Apache Hadoop* daemon 在被授予 Kerberos 化服务前必须通过 Kerberos 认证。Kerberos 化服务包括 MapReduce、HDFS 和 HBase。
比如，如果有一个包含 10 个节点的集群，其中 Primary NameNode 服务运行在 node1 上，DataNode 服务运行在 node5 上，则该 DataNode 服务在被允许和 Primary NameNode 服务通讯前，必须通过 Kerberos 认证。
- 可信任客户端 — 用户（包括人或系统）在集群上使用 Kerberos 化服务前必须通过 Kerberos 认证。不像其他简单认证，用户的 UID 不用来认证用户。而是将用户的 Kerberos 身份条目的基本组件作为用户名来认证。
比如，如果 Unix 用户 *jdoe* 需要运行 MapReduce 程序，则在程序被允许运行前，该用户需要通过 Kerberos 认证。而且，MapReduce 应用程序必须以认证用户的身份运行。

1.2 先决条件和推荐配置

要为集群设置 Kerberos 认证，你必须执行或了解以下事项：

- 英特尔® Apache Hadoop* 软件发行版已在 Kerberos 5 服务器上进行测试。进行测试的 Kerberos 5 服务器安装包是 Red Enterprise Linux 6.3 上的 krb5-server-1.9-33。由于厂家的应用各不相同，以及各版本之间的差异，英特尔® 发行版有可能（也有可能不能）在和测试环境不同的情况或版本下工作。但你也可能因此得到不一样的结果。
- 你必须了解 Kerberos 协议。
- 你必须了解什么是 Kerberos 身份条目（Principal），以及身份条目是由哪些组件构成的。
- 你必须了解什么是 Kerberos 领域（realm），以及它在 Kerberos 协议中是如何工作的。
- 你必须了解如何创建 Kerberos 密钥表（keytab）。
- 你必须对允许你创建密钥表和生成身份条目的密钥分发中心（KDC）有管理员权限。
- 所有 Apache Hadoop* 集群中的节点必须能通过 TCP 和 KDC 通讯，并且能用 Kerberos 客户端来生成 keytab。和集群通讯的 KDC 必须是你为 Apache Hadoop* 生成身份条目服务和客户端的 KDC。
- Apache Hadoop* 集群中的所有节点必须能和 KDC 的认证服务器（AS）通讯。
- 在集群的每个节点上，你必须为该节点上的 Apache Hadoop* 服务使用的 JDK 启用无限制的 JCE。
- 在 Apache Hadoop* 中，要么使用 Kerberos，要么什么都不用。如果 Kerberos 配置中有任何错误，则 Apache Hadoop* 服务将不能启动。由于配置 Kerberos 的复杂性，你很有可能在配置 Kerberos 中出错。因此，强烈推荐你首先配置一个使用简单认证的集群，并在配置 Kerberos 前



确认它能正常工作。这可以保证你之后在配置 Kerberos 遇到问题时，你可以确认这是 Kerberos 配置相关问题，而不是其它集群相关问题。

- 在配置 Kerberos 前，英特尔® Apache Hadoop* 软件发行版集群必须已安装配置，并且以简单认证模式在运行。更多信息参见英特尔® Apache Hadoop* 软件发行版安装手册。
- 考虑到 Apache Hadoop* 集群对于 KDC 和 AS 的需求，推荐你在集群中配置一个 KDC，专门为集群中的用户和服务进行服务。在这种情况下，你需要配置一个 AS，或 KDC 和已存在的 AS 之间的单向跨领域的信任关系。

1.3 为 Apache Hadoop* 配置 Kerberos

在英特尔® 发行版启用 Kerberos 之前，你需要执行一些 Kerberos 相关的步骤。以下章节阐述了这些步骤。

注释： 这些步骤假定你使用安装在 RHEL 操作系统上的 MIT Kerberos。

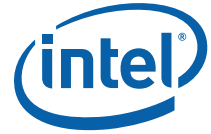
1.3.1 定义 Kerberos 领域

在 Kerberos 中，领域是一个管理域，在这个域中，认证服务器有权认证某个身份。领域用于确认一组相关身份条目。你必须为 Apache Hadoop* 集群定义 KDC 的领域。集群中的所有身份条目都将使用领域。要做到这点，执行以下步骤。

1. 登录到安装了 KDC 的机器。
2. 定义集群的领域。通常情况下，这就是用于解析集群中的节点主机名的 DNS 域。
3. 执行以下步骤对 `/etc/krb5.conf` 进行编辑。
 - a. 在文本编辑器中打开以下文件：`/etc/krb5.conf`
 - b. 在 `/etc/krb5.conf` 中，找到 `[realms]` 部分。
 - c. 在 `[realms]` 部分，找到以下内容：


```
EXAMPLE.COM = {
    kdc = kerberos.example.com
    admin_server = kerberos.example.com
}
```
 - d. 如果你找不到该内容，请自行创建。
 - e. 将 `EXAMPLE.COM` 替换为集群的领域。
 - f. 将 `kerberos.example.com` 替换为 KDC 的 FQDN。
 - g. 保存并关闭 `/etc/krb5.conf`。
4. 执行以下步骤对 `/var/kerberos/krb5kdc/kdc.conf` 进行编辑。
 - a. 在文本编辑器中打开以下文件 `/var/kerberos/krb5kdc/kdc.conf`。
 - b. 在 `/var/kerberos/krb5kdc/kdc.conf` 中，找到 `[realms]` 部分。
 - c. 在 `[realms]` 部分，找到以下内容：


```
EXAMPLE.COM = {
    #master_key_type = aes256-cts
    acl_file = /var/kerberos/krb5kdc/kadm5.acl
    dict_file = /usr/share/dict/words
    admin_keytab = /var/kerberos/krb5kdc/kadm5.keytab
    supported_encetypes = aes256-cts:normal aes128-cts:normal des3-
    hmac-sha1:normal arcfour-hmac:normal des-hmac-sha1:normal des-cbc-
    md5:normal des-cbc-crc:normal
}
```
 - d. 如果你找不到该内容，请自行创建。



- e. 将 EXAMPLE.COM 替换为集群的领域。
- f. 保存并关闭 `/var/kerberos/krb5kdc/kdc.conf`。
5. 执行以下步骤对 `/var/kerberos/krb5kdc/kadm5.acf` 进行编辑。
 - a. 在文本编辑器中打开以下文件: `/var/kerberos/krb5kdc/kadm5.acf`
 - b. 在文件中找到以下文本: `*/admin@EXAMPLE.COM`
 - c. 如果该内容不存在, 则自行创建。
 - d. 将 EXAMPLE.COM 替换为集群的领域。
 - e. 保存并关闭 `/var/kerberos/krb5kdc/kadm5.acf`。
6. 执行以下步骤, 创建领域的 KDC 数据库, 并设置 Kerberos 管理员密码。
 - a. 执行以下命令: `rm -f /etc/kadm5.keytab`
 - b. 执行以下命令:


```
kdb5_util -P <password> -r <realm> create -s
```

 其中:
 - `<password>` 是 KDC 数据库的密码。
 - `<realm>` 是 Apache Hadoop* 集群的领域。
 - c. 执行以下命令:


```
kadmin.local -q 'cpw -pw <admin_password> kadmin/admin'
```

 其中:
 - `<admin_password>` 是 kadmin/admin 的密码。

1.3.2 配置 Kerberos 化的 SSL

和 KDC 通讯, 交换 Kerberos 化的 SSL (KSSL) 加密的票据 (ticket)。由于 Java 6 中的某个故障, 使用了强加密运算法则的 Kerberos 票据不能在 KSSL 中使用。因此, 只有使用了弱加密运算法则的 Kerberos 票据才能在 Apache Hadoop* 中使用。以下步骤解释了如何配置 KDC, 以生成弱加密运算法则的票据。

警告: 通过配置用于生成弱加密运算法则的票据的 KDC, 理论上来说, 你降低了使用该 KDC 的系统的安全。

1. 登录到安装了 KDC 的机器。
2. 在文本编辑器中打开以下文件: `/etc/krb5.conf`
3. 在 `/etc/krb5.conf` 中, 找到 `[libdefaults]` 部分。
4. 在 `[libdefaults]` 部分, 找到以下属性: `allow_weak_crypto`
5. 如果属性不存在, 则在 `[libdefaults]` 部分创建。
6. 将 `allow_weak_crypto` 属性设为 true。比如:


```
allow_weak_crypto = true
```
7. 保存并关闭 `/etc/krb5.conf`。

1.3.3 安装和配置 Kerberos 客户端

如果客户端或服务要访问 Kerberos 化的服务, 它们必须通过 Kerberos 认证。如果客户端要通过 Kerberos 认证, 它需要有一个 Kerberos 客户端。在 Apache Hadoop* 集群中, 这表示需要在集群中的每个机器上安装和配置 Kerberos 客户端。要做到这一点, 在每个节点上执行以下步骤。

1. 安装 `krb5-workstation` rpm。
2. 章节 1.3.1 定义 Kerberos 领域和 1.3.2 配置 Kerberos 化的 SSL 提供了在 KDC 上编辑 `/etc/krb5.conf` 的指导说明。确认这些编辑已完成。
3. 从 KDC 上复制 `/etc/krb5.conf` 到节点上的 `/etc`。



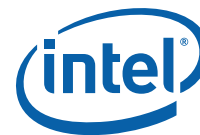
1.4 在 Intel® Manager 中更改安全模式

要让 Apache Hadoop* 集群使用 Kerberos 认证，你需要更改集群的安全模式。要做到这点，执行以下步骤。

1. 以管理员权限的用户登录 Intel® Manager for Apache Hadoop* software。
2. 确认没有 Apache Hadoop* 服务在运行中。
3. 在右上角，点击**配置向导**链接。
4. 在集群配置向导中，点击下一步按钮直至第 4 步配置安全模式的页面出现。在**安全策略**下拉菜单中，选择**集群中的节点使用 Kerberos 作为安全策略**。

The screenshot shows a web-based configuration window titled "第4步" (Step 4) with the subtitle "配置集群节点安全策略" (Configure cluster node security policy). The main instruction is "设置集群中节点使用的安全策略。" (Set the security policy used by the nodes in the cluster.). Below this, there is a label "安全策略：" (Security policy:) followed by a dropdown menu. The dropdown menu is open, showing the selected option: "集群中的节点使用Kerberos作为安全策略" (Cluster nodes use Kerberos as security policy). At the bottom right of the window, there are three buttons: "上一步" (Previous step), "下一步" (Next step), and "取消" (Cancel).

5. 继续集群向导，直至出现一个问你是否想要配置集群对话框。
6. 当被问到你是否要配置集群时，点击**取消**按钮。



1.5 创建密钥表

一旦安全模式被设置为 Kerberos，每个节点上将自动生成 Apache Hadoop*daemon 的身份条目。每个身份条目的密钥表必须由 KDC 生成，并上传到 Intel® Manager for Apache Hadoop* software，然后推送到集群中的每个节点。以下步骤演示如何做到这点。

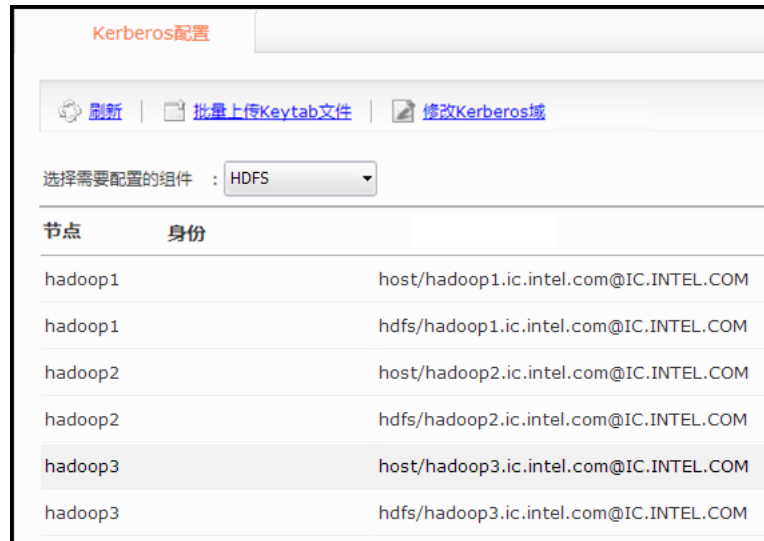
1. 执行以下步骤，确认身份条目已设置正确的领域。
 - a. 以管理员身份登录 Intel® Manager。
 - b. 在集群配置菜单中，选择 **Kerberos 配置** 选项。

节点	身份	更新Keytab时间	上传
hadoop1	zookeeper/hadoop1.ic.intel.com@IC.INTEL.COM	N/A	
hadoop2	zookeeper/hadoop2.ic.intel.com@IC.INTEL.COM	N/A	
hadoop3	zookeeper/hadoop3.ic.intel.com@IC.INTEL.COM	N/A	
hadoop1	host/hadoop1.ic.intel.com@IC.INTEL.COM	N/A	
hadoop1	hdfs/hadoop1.ic.intel.com@IC.INTEL.COM	N/A	
hadoop2	host/hadoop2.ic.intel.com@IC.INTEL.COM	N/A	
hadoop2	hdfs/hadoop2.ic.intel.com@IC.INTEL.COM	N/A	
hadoop3	host/hadoop3.ic.intel.com@IC.INTEL.COM	N/A	
hadoop3	hdfs/hadoop3.ic.intel.com@IC.INTEL.COM	N/A	
hadoop1	mapred/hadoop1.ic.intel.com@IC.INTEL.COM	N/A	
hadoop2	mapred/hadoop2.ic.intel.com@IC.INTEL.COM	N/A	
hadoop3	mapred/hadoop3.ic.intel.com@IC.INTEL.COM	N/A	

- c. 如果领域不正确，则点击**修改 Kerberos 域**按钮。
 - d. 在修改 Kerberos 域对话框，在**新 Kerberos 域**区域输入正确的域，然后点击**确定**按钮。
2. 以 root 用户登录到安装了 KDC 的机器上。
3. 确认 KDC 在运行中
4. 确认你对 KDC 有管理员权限的访问。
5. 创建一个名为 `/tmp/hadoop_keytabs` 的目录。
6. 在 `/tmp/hadoop_keytabs` 中，为集群中的每个节点分别创建一个目录。目录名必须是节点的 FQDN。
7. 使用 `cd` 命令进入某个节点的目录。
8. 对于这一节点，你必须确认你需要生成密钥表文件的所有身份条目。要做到这点，执行以下步骤。
 - a. 以管理员身份登录 Intel® Manager。
 - b. 在集群配置菜单中，选择 **Kerberos 配置** 选项。



- c. 在 Kerberos 配置页面，从**选择需要配置的组件**下拉菜单中选择 HDFS。



- d. 对于 HDFS 服务，确认该节点的所有身份条目。
- e. 使用**选择需要配置的组件**下拉菜单来确认该节点的所有其他服务。
9. 回到 `/tmp/hadoop_keytabs` 中的节点目录。
10. 执行以下命令，为每个服务添加身份条目：
- ```
kadmin.local -w secure -q 'addprinc -randkey <PRINCIPAL>'
```
11. 执行以下命令，为每个身份条目生成密钥表：
- ```
kadmin.local -w secure -q 'xst -k <KEYTAB FILE>.keytab <PRINCIPAL>'
```
- <KEYTAB FILE> 必须是生成身份条目对应的服务名称。比如，如果你为 HDFS 生成一个密钥表，则文件名必须为 `hdfs.keytab`。
12. 为集群中的每一个节点重复以上步骤，密钥表文件将在对应的节点目录生成。

1.5.1 生成密钥表示例

以下情形演示了如何创建密钥表文件。在此情形中，具体如下：

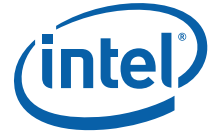
- 这是一个包含三个节点的集群。
- 每个节点的 FQDN 是：hadoop1.example.com, hadoop2.example.com 和 hadoop3.example.com。
- DNS 域名为 example.com，领域名为 EXAMPLE.COM。
- 集群上只安装了 HDFS 和 MapReduce 服务。

要在此情形中创建密钥表，执行以下操作。

1. 在 KDC 运行的节点上，创建以下目录：
 - `/tmp/hadoop_keytabs`
 - `/tmp/hadoop_keytabs/hadoop1.example.com`
 - `/tmp/hadoop_keytabs/hadoop2.example.com`
 - `/tmp/hadoop_keytabs/hadoop3.example.com`
2. 执行以下步骤，为 hadoop1 创建密钥表和身份条目。



- a. 通过 cd 命令进入 `/tmp/hadoop_keytabs/hadoop1.example.com`
 - b. 执行以下命令，为 HDFS 创建一个主身份条目。
`kadmin.local -w secure -q 'addprinc -randkey host/hadoop1.example.com@EXAMPLE.COM'`
 - c. 执行以下命令，为主身份条目创建一个密钥表。
`kadmin.local -w secure -q 'xst -k host.keytab host/hadoop1.example.com@EXAMPLE.COM'`
 - d. 执行以下命令，为 HDFS 创建一个 HDFS 身份条目。
`kadmin.local -w secure -q 'addprinc -randkey hdfs/hadoop1.example.com@EXAMPLE.COM'`
 - e. 执行以下命令，为 HDFS 身份条目创建一个密钥表。
`kadmin.local -w secure -q 'xst -k host.keytab hdfs/hadoop1.example.com@EXAMPLE.COM'`
 - f. 执行以下命令，为 HDFS 创建一个 MapReduce 身份条目。
`kadmin.local -w secure -q 'addprinc -randkey mapred/hadoop1.example.com@EXAMPLE.COM'`
 - g. 执行以下命令，为 MapReduce 身份条目创建一个密钥表。
`kadmin.local -w secure -q 'xst -k mapred.keytab hdfs/hadoop1.example.com@EXAMPLE.COM'`
3. 执行以下步骤，为 hadoop2 创建密钥表和身份条目。
- a. 通过 cd 命令进入 `/tmp/hadoop_keytabs/hadoop2.example.com`
 - b. 执行以下命令，为 HDFS 创建一个主身份条目。
`kadmin.local -w secure -q 'addprinc -randkey host/hadoop2.example.com@EXAMPLE.COM'`
 - c. 执行以下命令，为主身份条目创建一个密钥表。
`kadmin.local -w secure -q 'xst -k host.keytab host/hadoop2.example.com@EXAMPLE.COM'`
 - d. 执行以下命令，为 HDFS 创建一个 HDFS 身份条目。
`kadmin.local -w secure -q 'addprinc -randkey hdfs/hadoop2.example.com@EXAMPLE.COM'`
 - e. 执行以下命令，为 HDFS 身份条目创建一个密钥表。
`kadmin.local -w secure -q 'xst -k host.keytab hdfs/hadoop2.example.com@EXAMPLE.COM'`
 - f. 执行以下命令，为 HDFS 创建一个 MapReduce 身份条目。
`kadmin.local -w secure -q 'addprinc -randkey mapred/hadoop2.example.com@EXAMPLE.COM'`
 - g. 执行以下命令，为 MapReduce 身份条目创建一个密钥表。
`kadmin.local -w secure -q 'xst -k mapred.keytab hdfs/hadoop2.example.com@EXAMPLE.COM'`
4. 执行以下步骤，为 hadoop3 创建密钥表和身份条目。
- a. 通过 cd 命令进入 `/tmp/hadoop_keytabs/hadoop3.example.com`
 - b. 执行以下命令，为 HDFS 创建一个主身份条目。
`kadmin.local -w secure -q 'addprinc -randkey host/hadoop3.example.com@EXAMPLE.COM'`
 - c. 执行以下命令，为主身份条目创建一个密钥表。
`kadmin.local -w secure -q 'xst -k host.keytab host/hadoop3.example.com@EXAMPLE.COM'`
 - d. 执行以下命令，为 HDFS 创建一个 HDFS 身份条目。



- ```
kadmin.local -w secure -q 'addprinc -randkey hdfs/
hadoop3.example.com@EXAMPLE.COM'
```
- e. 执行以下命令，为 HDFS 身份条目创建一个密钥表。  

```
kadmin.local -w secure -q 'xst -k host.keytab hdfs/
hadoop3.example.com@EXAMPLE.COM'
```
  - f. 执行以下命令，为 HDFS 创建一个 MapReduce 身份条目。  

```
kadmin.local -w secure -q 'addprinc -randkey mapred/
hadoop3.example.com@EXAMPLE.COM'
```
  - g. 执行以下命令，为 MapReduce 身份条目创建一个密钥表。  

```
kadmin.local -w secure -q 'xst -k mapred.keytab hdfs/
hadoop3.example.com@EXAMPLE.COM'
```

## 1.6 在 Apache Hadoop\* 集群中部署密钥表

要在 Apache Hadoop\* 集群中部署密钥表 s，执行以下步骤。

1. 执行章节 1.6 在 Apache Hadoop\* 集群中部署密钥表中的步骤。
2. 创建一个 `/tmp/hadoop_keytabs` 的 tarball 。
3. 将 tarball 复制到你用于访问 Intel® Manager for Apache Hadoop\* software 的机器。
4. 以管理员身份登录 Intel® Manager。
5. 在集群配置菜单中，选择 **Kerberos 配置** 选项 。
6. 在 Kerberos 配置页面，点击**批量上传 Keytab 文件**链接。
7. 在批量上传 Keytab 文件对话框，点击**选择**按钮。
8. 在打开对话框，浏览并双击 `hadoop_keytabs` 这一 tarball 文件。
9. 在上传 Keytab 文件对话框，点击**上传**按钮，然后点击**关闭**按钮。
10. 在集群配置菜单中，点击**集群节点**子页面。
11. 在集群节点页面，点击**配置所有节点**链接。
12. 执行以下步骤，重启 Intel® Manager 服务。
  - a. 以 root 权限的用户登录到管理节点。
  - b. 执行以下命令：`service intel-manager restart`

## 1.7 在安全模式中测试 Apache Hadoop\* 功能

在 Apache Hadoop\* 集群启用 Kerberos 后，你需要确保 Kerberos 配置正常工作。如果 Kerberos 配置错误，则集群服务可能无法启动或不能被客户端访问。要验证 Apache Hadoop\* 在安全模式中正常运行，执行以下步骤。

1. 以管理员身份登录 Intel® Manager for Apache Hadoop\* software。
2. 在集群概况菜单，选择**控制面板**选项。
3. 在控制面板页面，点击**集群概述**子页面。
4. 在集群概述子页面，启动所有的服务。
5. 确认所有的服务都已成功启动。
6. 登录到有访问 MapReduce 和 HDFS 客户端的机器。
7. 如果你还没这么做，通过 `su` 命令切换到一个非 root、但对这些客户端有访问权限的 Unix 用户。
8. 如果你还没这么做，通过 `kinit` 来获得并缓存一个用户身份条目的初始授权票据。
9. 对非 root 用户，使用 HDFS 客户端访问 HDFS，使用 MapReduce 客户端访问 MapReduce。



10. 如果集群中安装了 Hive 和 HBase，请查看用户是否能访问这些服务。

## 1.8 解决 Kerberos 配置问题

### 1.8.1 客户端访问失败并显示错误信息 SASL authentication failed

安全模式被启用后，当客户端尝试访问某个服务，比如 HDFS 或 MapReduce 时，以下错误信息可能显示。

```
ERROR: java.lang.RuntimeException:SASL authentication failed.The most likely cause is missing or invalid credentials.Consider 'kinit'.
```

要和 Hadoop 集群中的服务通讯，用户必须具备有效的 Kerberos 票据。如果用户不具备有效的 Kerberos 票据，你将看到认证失败。

要解决这一问题，用户需要生成正确凭据和票据的密钥表文件。要做到这一点，用户需要执行以下命令：

```
kinit -k -t <keytab_file> <principal_name>@<realm>
```

### 1.8.2 客户端访问失败并显示错误信息 GSS initiate Failed

在 Kerberos 版本 1.81 或更高版本中，当用户尝试和 Apache Hadoop\* 集群在安全模式下通讯时，以下出错信息可能会在 Kerberos 认证时出现。

```
java.io.IOException: javax.security.sasl.SaslException:GSS initiate failed
```

出错的原因在于版本 1.8.1 中 Kerberos 票据缓存的创建。Java 6 Update 26 或更低版本读取票据缓存失败。

要解决这一问题，执行以下步骤。

1. 对用户生成具有正确的凭据和票据的密钥表文件。要做到这一点，用户需要执行以下命令：

```
kinit -k -t <keytab_file> <principal_name>@<realm>
```

2. 执行以下命令，生成 JRE 可读取的票据缓存格式。

```
kinit -R
```



## 2.0 为 Apache Hadoop\* 服务设置基于角色的验证

验证是确认客户端身份的一个过程，或者说，是用来确保客户端是本人的一个过程。验证是决定被验证的身份是否有权使用特定资源的一个过程。

如果是在简单模式下，Apache Hadoop\* 根据 Unix Shell 中的用户名对用户进行验证，如果是在安全模式下，则根据用户的 Kerberos 身份条目进行验证。集群使用访问控制列表（ACL）来执行服务级别的验证。ACL 是一个允许对 Apache Hadoop\* 服务执行特定操作的用户和组的列表。

用户被验证后，Apache Hadoop\* daemon 会检查用户是否被列在访问控制列表，或是否属于列在访问控制列表的某个组。如果二者之一符合，则用户将被授权执行访问控制列表控制的操作。英特尔® Apache Hadoop\* 软件发行版使用基于角色的验证工具，此工具使用基于 LDAP 的组映射功能。以下章节解释了如何使用这一功能。

### 2.1 功能和作用

基于角色的验证提供以下功能和作用：

- 统一安全管理工具 — 在 Intel® Manager 中，系统管理员使用中央管理控制点定义角色、分配用户和组给这些角色，然后通过点击某个按钮将这些设置部署到集群中。
- 与已存在的身份 store 集成 — 你不必从零开始重新建设 identity store，基于角色的验证功能可让你从已存在的 Active Directory\* (AD) 获取用户和组信息，然后在 Intel® Manager 中将这身份与角色绑定。
- 安全性、一致性以及可靠性 — 由于 Intel® Manager 从单个 AD 获取和分配用户和组，系统管理员避免了当多个 identity store 被使用和访问控制列表在集群中被手工部署这些情况下可能存在的访问控制列表和权限漏洞。

### 2.2 访问控制列表是如何工作的？

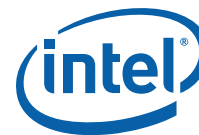
一旦客户在 Shell 或 Kerberos 身份条目中通过 Unix 用户名被验证，则 Apache Hadoop\* daemon 执行以下访问控制列表的检查：

1. 如果访问控制列表为空，则没有组或用户可执行操作。
2. 如果访问控制列表包含星号，则任何组或用户可执行操作。
3. 如果客户端的用户名被列在某服务的访问控制列表中，则客户端允许访问该服务。
4. 根据用户名，daemon 将检查 LDAP 服务器，以决定用户所属 LDAP 组是否已被列入访问控制列表中。如果是，则用户被授权访问。
5. 如果客户端的用户名或组没有在访问控制列表中找到，则服务将拒绝客户端的访问请求。

### 2.3 什么是角色？

角色是创建、读取和编辑数据的一系列权限，以及对以下 Apache Hadoop\* 服务的管理权限：

- HDFS
- MapReduce
- HBase



- Hive
- Oozie

角色可有一个或多个服务的权限。角色被分配给某个 AD 组后，此组中的任何用户可获得该角色拥有的一系列权限。如果用户位于多个 AD 组，且每个 AD 组被分配了一个角色，则该用户可获得被赋予给这些组的所有角色的权限。而且，你可以将该用户在某个 AD 组中添加或删除，轻松实现该用户对 Apache Hadoop\* 服务的访问权限的增加或减少。

基于角色的验证的启用根据不同的服务而定。一旦启用，用户必须有一个拥有合适权限的角色，用于执行服务有关的所有读 / 写操作。角色可以有一个、多个或所有以下权限：

表 1. 角色的权限

| 服务        | 权限                      | 描述                                                                                                                                                                                                                                                                                                             |
|-----------|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HDFS      | HDFS 文件访问               | 能在 HDFS 文件系统读取、编辑和创建文件。这一权限不会取代已有的 HDFS 文件权限，因此如果用户或用户所在的组不能读取或写入该文件，则 <i>HDFS 文件访问</i> 权限不能授予该文件的读 / 写访问权限。<br>如果 HDFS 已启用基于角色的验证，则用户必须具备拥有这一权限的角色，或具备访问一个或多个服务的权限，比如 MapReduce。<br><b>注释：</b> 如果表中的任何其他权限被分配给某个角色，则表示该角色给了用户在 HDFS 文件系统上创建、编辑文件的权限，即使 <i>HDFS 文件访问</i> 权限并没有授予给该角色。                          |
| HDFS      | 访问这些指定的文件               | 授予写、读和执行权限给用户定义的文件或文件夹列表。 <i>访问这些指定的文件</i> 权限将取代已有的 HDFS 文件权限，因此如果文件系统权限之间存在冲突，则 <i>访问这些指定的文件</i> 权限将被优先应用。<br>比如，如果 <i>访问这些指定的文件</i> 权限授予用户对文件 <i>/user/root/a.txt</i> 的 <i>jdoe</i> 读权限，则 <i>jdoe</i> 能读取该文件，即使该文件所有者为其他用户且文件权限为 600。<br><i>访问这些指定的文件</i> 权限允许用户在 HDFS 文件系统上读取、编辑和创建那些未被列入用户定义的文件和文件夹列表上的文件。 |
| HDFS      | 管理 HDFS 系统，成为 DFSAdmin  | 能执行 <i>dfsadmin</i> 命令。这一权限不允许用户执行 <i>dfsadmin</i> 子命令，比如 <i>report</i> ，这需要超级用户的权限。                                                                                                                                                                                                                           |
| MapReduce | 提交 Job 到指定的队列           | 能提交作业到用户定义的 MapReduce 队列列表中。如果队列不在列表中，则角色不被授予提交作业到队列的权限。<br>如果 MapReduce 已启用基于角色的验证，则用户不能提交作业到队列中，除非用户具备某个带此权限的角色。                                                                                                                                                                                             |
| MapReduce | 管理指定队列中的 Job            | 某个用户和组能执行以下 MapReduce 队列中用户定义的列表上的作业有关操作。 <ul style="list-style-type: none"> <li>• 中断一个作业</li> <li>• 中断一个作业的任务</li> <li>• 取消一个作业的任务</li> <li>• 设置作业的优先级</li> </ul> 无论基于角色的验证如何设置，作业的所有者可能需要经常执行以上操作。                                                                                                           |
| MapReduce | 管理 MapReduce，成为 MRAdmin | 能执行 <i>mradmin</i> 命令。这一权限不允许用户执行 <i>mradmin</i> 子命令，这需要超级用户的权限。                                                                                                                                                                                                                                               |



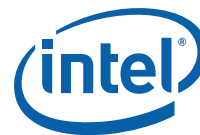


表 1. 角色的权限

| 服务    | 权限          | 描述                                                                                                                                                                                                                                                                      |
|-------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HBase | 读取数据        | <p>能读取一个或多个表中的所有列簇，并且 / 或者读取一个或多个表中的一些列簇。这些允许用户读取的列簇是基于用户定义的列表的。如果一个或多个表被列入列表中，则用户或组被允许读取一个或多个表中的所有列簇。</p> <p>这一权限仅允许用户或组查看列表中的列簇。如果表或列簇不在列表中，则这一权限不允许用户或组读取列簇。</p> <p>如果 HBase 已启用基于角色的验证，则用户不能读取列簇，除非用户有带此权限的角色，或者用户属于某个带此权限的角色的组。</p>                                |
| HBase | 写入数据        | <p>能更新或插入数据到一个或多个表的所有列簇中，并且 / 或者更新或插入数据到一个或多个表的一些列簇中。此权限不允许用户或组在表中创建列簇。</p> <p>权限允许用户编辑的列簇是基于用户定义的列表的。如果列表中仅指定一个或几个表，则权限允许用户或组编辑该表或这几个表。</p> <p>权限仅允许用户或组编辑列表中的列簇。如果表或列簇不在列表中，则此权限不允许用户或组编辑列簇。</p> <p>如果 HBase 已启用基于角色的验证，则用户不能编辑列簇，除非用户有带此权限的角色，或者用户属于某个带此权限的角色的组。</p> |
| HBase | 管理维护        | <p>用户或组能对用户定义列表中的表执行以下操作：</p> <ul style="list-style-type: none"> <li>• 更改表属性。</li> <li>• 添加、更改或删除列簇。</li> <li>• 启用、禁用或删除表。</li> <li>• 触发 region (re)assignments 或 relocation。</li> </ul> <p>用户定义列表中应仅包含表。列表中不能包含任何列簇。</p>                                               |
| HBase | 创建列簇        | <p>允许用户或组在用户定义列表的表中创建列簇。</p> <p>此权限允许用户或组创建列表中的列簇。如果表不在列表中，则此权限不允许用户或组在该表中创建列簇。</p> <p>如果 HBase 已启用基于角色的验证，则用户不能在表中创建列簇，除非用户有带此权限的角色，或者用户属于某个带此权限的角色的组。</p>                                                                                                             |
| HBase | 超级用户        | <p>允许用户或组创建表并将 balancer 打开或关闭。同时，允许读取 / 写入数据库中的所有表和列簇。</p>                                                                                                                                                                                                              |
| Hive  | 查看数据库       | <p>允许用户或组查看 Hive 中的所有数据库</p>                                                                                                                                                                                                                                            |
| Hive  | 在数据库或表中创建对象 | <p>允许用户或组在用户定义列表的表和 / 或数据库中创建对象。</p> <p>这些允许用户创建的对象是基于用户定义的列表的。如果只有数据库被列入列表中，则用户或组被允许在该数据库中创建对象，包括表，以及在数据库的表中创建对象。</p> <p>此权限允许用户或组在列表中的表和数据库中创建对象。如果表或数据库不在列表中，则此权限不允许用户或组在该数据库或表中创建对象。</p> <p>如果 Hive 已启用基于角色的验证，则用户不能在数据库或表中创建对象，除非用户有带此权限的角色，或者用户属于某个带此权限的角色的组。</p> |



表 1. 角色的权限

| 服务    | 权限                      | 描述                                                                                                                                                                                                                                                                                                                                                                                            |
|-------|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Hive  | 更新数据库或者表中的数据            | <p>允许用户或组在用户定义列表的表和 / 或数据库中更新数据。这些允许用户更新的数据是基于用户定义的列表的。如果只有数据库被列入列表中，则用户或组被允许在该数据库中任何地方更新数据，包括数据库的表。</p> <p>此权限允许用户或组在列表中的表和数据库中更新数据。如果表或数据库不在列表中，则此权限不允许用户或组在该数据库或表中更新数据。</p> <p>如果 Hive 已启用基于角色的验证，则用户不能在数据库或表中更新数据，除非用户有带此权限的角色，或者用户属于某个带此权限的角色的组。</p>                                                                                                                                     |
| Hive  | 修改数据库或者表的结构             | <p>允许用户或组在用户定义列表的表和 / 或数据库中更改元数据。这些允许用户更改的元数据是基于用户定义的列表的。如果只有数据库被列入列表中，则用户或组被允许在该数据库中任何地方更改元数据，包括数据库的表。</p> <p>此权限允许用户或组在列表中的表和数据库中更改元数据。如果表或数据库不在列表中，则此权限不允许用户或组在该数据库或表中更改元数据。</p> <p>如果 Hive 已启用基于角色的验证，则用户不能在数据库或表中更改元数据，除非用户有带此权限的角色，或者用户属于某个带此权限的角色的组。</p>                                                                                                                               |
| Hive  | 删除数据库或者表                | <p>允许用户或组在删除用户定义列表的表和 / 或数据库。这些允许用户删除的表和数据库是基于用户定义的列表的。如果只有数据库被列入列表中，则用户或组被允许在该数据库中删除任何表。</p> <p>此权限允许用户或组删除列表中的表和数据库。如果表或数据库不在列表中，则此权限不允许用户或组删除数据库或表。</p> <p>如果 Hive 已启用基于角色的验证，则用户不能删除数据库或表，除非用户有带此权限的角色，或者用户属于某个带此权限的角色的组。</p>                                                                                                                                                             |
| Oozie | 被其他用户代理 (impersonation) | <p>当运行 oozie 流程时，允许用户被另一用户代理。比如，如果用户 <i>jdoue</i> 被赋予带此权限的角色，用户 <i>gsmith</i> 被赋予带 <i>被其他用户代理</i> 权限的角色，则用户 <i>gsmith</i> 能够像用户 <i>jdoue</i> 一样执行 oozie 流程。</p> <p>如果带有此权限的角色被分配给某个 AD 组，则该 AD 组的任何成员都能被其他用户代理。</p> <p>要让其他用户或组代理，这一权限是必须的但不是足够的。而且，想要代理的用户或组必须被赋予 <i>代理其他用户</i> 权限，而且此权限必须明确定义哪些角色可被代理。</p>                                                                                  |
| Oozie | 代理其他用户                  | <p>在运行 oozie 流程时，允许用户代理其他用户。比如，如果用户 <i>gsmith</i> 被赋予带此权限的角色，用户 <i>jdoue</i> 被赋予带 <i>被其他用户代理</i> 权限的角色，则用户 <i>gsmith</i> 能够像用户 <i>jdoue</i> 一样执行 oozie 流程。</p> <p>如果带有此权限的角色被分配给某个 AD 组，则该 AD 组的任何成员都能模拟特定用户的身份。用户必须被赋予带有 <i>被其他用户代理</i> 权限的角色，而且这些角色必须在 <i>代理其他用户</i> 列表中被定义。</p> <p>要代理其他用户或组，这一权限是必须的但不是足够的。被代理的用户必须被赋予带有 <i>被其他用户代理</i> 权限的角色。而且，被模拟身份的用户或组必须在 <i>代理其他用户</i> 权限中明确定义。</p> |
| Oozie | 管理员                     | 允许用户或组执行 Oozie 管理操作。                                                                                                                                                                                                                                                                                                                                                                          |



## 2.4 先决条件

要使用基于角色的验证，你必须知道或操作以下事项：

- 基于角色的验证功能仅在 Active Directory 上测试过。如果你和其他基于 LDAP 的 identity store 集成，结果将不可预测。
- 你需要在 Active Directory 中设置中正确的用户和组，以验证对 Apache Hadoop\* 集群的资源访问。
- 所有 Apache Hadoop\* 集群中的节点必须能通过 TCP 和 AD 服务器绑定。
- 在简单认证模式中，用户需要有某一节点的 Unix 账户，其中 Apache Hadoop\* 的 daemon 和客户端必须允许用户执行这些服务的命令。
- 在 Kerberos 认证模式中，由于用户只需要一个 Kerberos 身份条目，用户不需要有集群中某个节点的 Unix 账户。身份条目是使用 kinit 命令登录而生成的。在使用 kinit 命令后，Apache Hadoop\* daemon 将从 Kerberos 缓存中获取用户名，而不是登录 shell。  
但是，有一个例外。如果用户需要运行 MapReduce 作业，则用户需要一个 Unix 账户，因为 MapReduce 作业的运行必须在相应的 Unix 用户名下启动作业进程。
- 要根据用户所在 AD 组来验证用户，用户必须存在于某个 AD。比如，如果 AD 用户 *Chris Lee* 属于 AD 组 *hadoop-admins*，且 AD 用户的登录名是 *clee*，用户被授予 Apache Hadoop\* 资源的访问权限，由于 *Chris Lee* 属于 *hadoop-admins* 组，这表示某个名为 *clee* 的 Unix 用户存在于集群中的某个节点，或存在于 Kerberos 缓存中。

## 2.5 设置和 Active Directory 服务器的通讯

当 Apache Hadoop\* daemon 的用户访问列表被启用，daemon 可查询 AD 服务器以决定用户是否在某个 AD 组 and 用户访问列表上。如果用户在这样的 AD 组，则用户被授权用户访问列表允许的操作。对于和工作映射的 LDAP 组，你必须在 Intel® Manager for Apache Hadoop\* software 中配置一个 identity store，以包含和 AD 服务器绑定的必要信息。当某个 Unix 用户和 AD 组进行映射时，集群中的每个节点的每个 Apache Hadoop\* daemon 使用这一信息与 AD 服务器绑定。

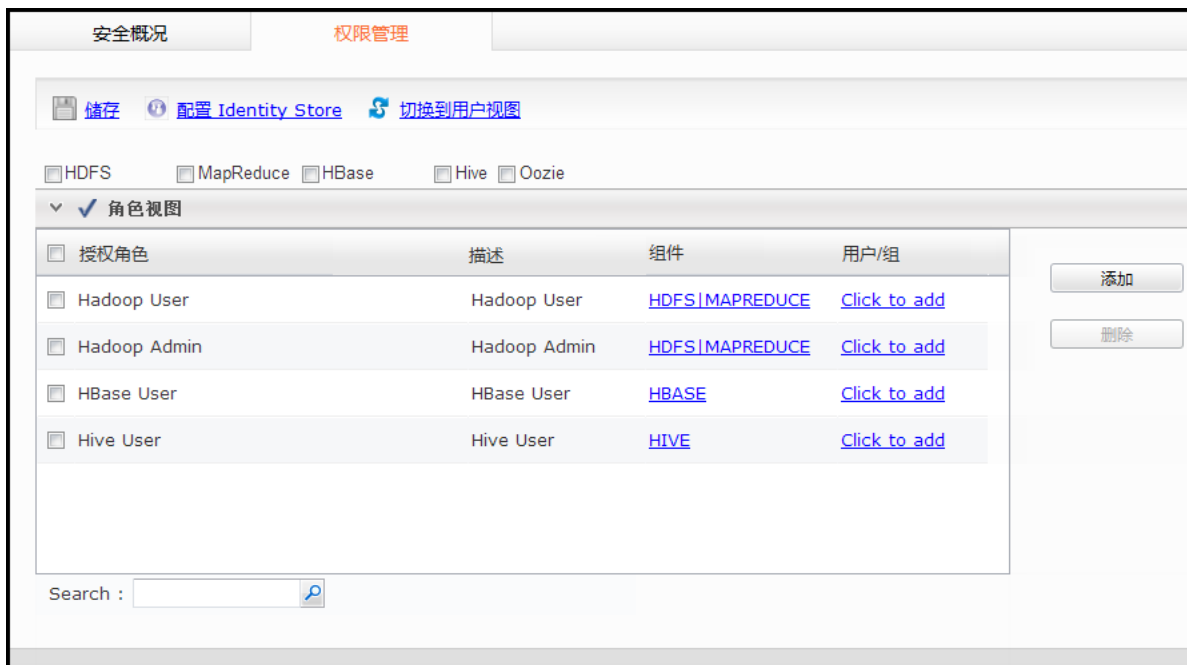
要设置绑定到 AD 的 identity store，执行以下步骤。

1. 获得以下关于 AD 的服务器信息：
  - AD 服务器的主机名和 TCP 端口
  - daemon 绑定 LDAP 服务器时使用的 DN 和密码。

**注意：** 绑定的 DN 应该对 LDAP 服务器仅有读的权限。

- LDAP 连接的查找起点。查找起点是一个标识名（DN），通常为 LDAP 根目录，即你开始查找的地方。
2. 以管理员身份登录 Intel® Manager。
  3. 在集群配置菜单，点击**安全**选项。

4. 在安全菜单，点击**权限管理**子页面。

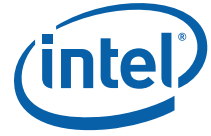


5. 在权限管理菜单，点击**配置 Identity Store** 链接。



6. 在**服务器**栏内，输入 AD 服务器的 FQDN。

7. 在**端口**栏内，输入 AD 服务器用于绑定请求的侦听 TCP 端口。



8. 在**绑定 DN** 栏，输入 Admin 用户标识名，此用户需有权读取基于角色的验证中使用的所有 AD 用户和组。

注意：

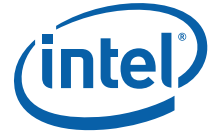
绑定 DN 栏内输入的标识名应对 AD 服务器仅有读的权限。

9. 在**绑定 DN 密码**栏内，输入**绑定 DN** 栏内指定的标识名的 AD 密码。
10. 在**查找起点** 栏内，输入一个标识名以指定 AD 某个点，即你想要 LDAP 开始查找 AD 组的地方。
11. 在**用户 ID 属性**栏内，输入 `sAMAccountName`。
12. 在**组 ID 属性**栏内，输入 `sAMAccountName`。
13. 点击**测试**按钮。然后，出现一个对话框，显示和 AD 服务器绑定成功的消息。
14. 关闭对话框，然后点击**保存**按钮。
15. 在权限管理子页面，点击**储存**链接。

## 2.6 在集群中部署基于角色的验证变化

当基于角色的验证在 Intel® Manager for Apache Hadoop\* software 中被配置或更改，这并不表示集群中的任何节点会自动开始执行基于角色的验证。一旦验证被启用、禁用或编辑，你必须执行以下步骤以在集群中部署这些变化，然后集群中的每个节点才会开始强制实现你配置的基于角色的验证。

1. 登录到 Intel® Manager。
2. 确认 HBase 组件已停止。如果没有，则立即停止。
3. 在集群配置菜单，点击**安全**选项。
4. 在安全页面，点击权限管理子页面。
5. 在权限管理子页面，执行以下其中之一或全部操作：
  - 创建一个角色。
  - 添加用户 / 组到角色。
  - 从角色中删除用户 / 组。
  - 配置 identity store。
  - 为其中一个服务启用或禁用基于角色的验证。
  - 添加用户或组。
6. 在权限管理子页面，点击**储存**链接。
7. 在集群配置菜单中，点击**集群节点**子页面。
8. 在集群节点子页面，点击配置所有节点链接。
9. 在确认对话框，点击**确定**。
10. 如果你为其中一个服务启用或禁用了基于角色的验证，则 Intel® Manager 可能重启。如果发生这一情形，重启后请重新登录到 Intel® Manager。
11. 在 Intel® Manager 中，执行以下其中之一或全部步骤：
  - 如果你为组件启用或禁用了基于角色的验证，你需要重启该组件。
  - 如果你添加或删除了角色的 HBase 超级用户权限，你需要重启 HBase 组件。
  - 如果你添加或删除了角色的 Oozie 权限，你需要重启 Oozie 组件。



## 2.7 控制 LDAP 组的缓存

默认情况下，当 Apache Hadoop® daemon 为用户从 AD 中获取 LDAP 组时，服务会将这些组缓存在内存中 5 分钟。一旦组被缓存，daemon 在缓存失效前将不再检查 AD 来决定用户是否在这些组中。而且，如果用户在 AD 组中被删除或添加，你可能需要等待至少 5 分钟以使集群中基于角色的验证适应这些变化。

在某些情况下，根据你所在企业的安全策略，这一缓存失效时间可能过长。在另一些情况下，缓存失效时间可能太短，因为来自集群的 LDAP 请求对 AD 来说数量过多。以下步骤演示了如何调整缓存失效时间。

1. 以管理员身份登录到 Intel® Manager。
2. 在集群配置菜单中，选择 **Hadoop** 子页面。
3. 在 Hadoop 页面，点击**全配置**子页面。
4. 在全配置子页面，点击**添加**按钮。
5. 在属性编辑对话框，执行以下步骤。
  - a. 在**属性**栏内，输入 **hadoop.security.groups.cache.secs**。
  - b. 在**默认值**栏内，输入为用户缓存 LDAP 组的秒数。
  - c. 点击**确定**。
6. 在全配置页面，点击**储存**链接。
7. 要将这些变化部署到集群中的每个节点，执行章节 2.6 在集群中部署基于角色的验证变化中的步骤。
8. 在 Intel® Manager 中，重启所有组件。

## 2.8 基于角色验证的用户场景

由于基于角色的验证具体实施中的复杂性和多样性，我们不能提供一个让所有用户都能遵循的统一步骤。每个基于角色的验证的具体实施对于实施它的组织来说都是独一无二的。以下用户场景演示了如何在该情形中实施基于角色的验证。你可以参考这一场景中的信息和步骤，来实施你自己的基于角色的验证。

Acme 公司部署了一个英特尔® Apache Hadoop® 软件发行版 集群。集群包含以下服务：HDFS、MapReduce、Hive 和 HBase。公司具有以下资源：

- 名称为 *Development* 的 MapReduce 队列。公司使用公平调度器（fair scheduler）。
- 名为 *default* 的数据库中有一个名为 *warehouse* 的 Hive 表。
- HBase 表的名为 *phone\_records*。

公司想要对集群资源机架开展以下细粒度控制：

- 谁被允许提交作业到 MapReduce *development* 队列。
- 谁被允许读取和写入 Hive *warehouse* 表。
- 谁被允许读取和写入 HBase *phone\_records* 表。
- 谁有 MapReduce、Hive、HDFS 和 HBase 的管理员或超级用户访问权限。



### 2.8.1 定义角色

根据章节 2.3 什么是角色？中为角色定义的权限，Acme 定义了以下角色，这些角色需要在 Intel® Manager 中创建。

表 2. 用户场景的角色

| 角色                             | 服务        | 权限                                                                                                                                                                 |
|--------------------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| access_mapredqueue_development | MapReduce | <ul style="list-style-type: none"> <li>提交作业到 development 队列</li> </ul>                                                                                             |
| access_hbasetable_phonerecords | HBase     | <ul style="list-style-type: none"> <li>读取表、列簇 phone_records</li> <li>写入表、列簇 phone_records</li> </ul>                                                               |
| access_hivetable_warehouse     | Hive      | <ul style="list-style-type: none"> <li>在数据库中创建对象和 warehouse 表。</li> <li>更新数据库中数据或 warehouse 表。</li> </ul>                                                          |
| hadoop-superusers              | HDFS      | <ul style="list-style-type: none"> <li>管理 HDFS 系统，成为 DFSAdmin</li> </ul>                                                                                           |
|                                | MapReduce | <ul style="list-style-type: none"> <li>提交作业到 development 队列</li> <li>development 队列的管理员作业</li> <li>管理 MapReduce，成为 MRAdmin</li> </ul>                              |
|                                | HBase     | <ul style="list-style-type: none"> <li>超级用户</li> </ul>                                                                                                             |
|                                | Hive      | <ul style="list-style-type: none"> <li>在数据库或表中创建默认数据库的对象</li> <li>更新数据库中的数据，或默认数据库中的表</li> <li>查看数据库</li> <li>更改数据库的结构，或默认数据库的表</li> <li>删除 warehouse 表</li> </ul> |

### 2.8.2 分配角色给用户

Acme 公司已定义以下用户，并参照章节 2.8.1 定义角色根据每个用户的职责分配了角色。

表 3. 用户场景的角色分配

| AD 显示名       | AD 登录名 | 职责                                          | 角色                                                                                                                   |
|--------------|--------|---------------------------------------------|----------------------------------------------------------------------------------------------------------------------|
| Chris Lee    | clee   | 为 HDFS、MapReduce、HBase 和 Hive 提供管理、操作和系统支持。 | hadoop-superusers                                                                                                    |
| George Smith | gsmith | MapReduce 开发者，即设计和测试 MapReduce 应用程序的人员。     | access_mapredqueue_development                                                                                       |
| John Doe     | jdoe   | 数据库开发者，即创建和编辑 Hive 和 Hbase 表的人员。            | <ul style="list-style-type: none"> <li>access_hivetable_warehouse</li> <li>access_hbasetable_phonerecords</li> </ul> |

### 2.8.3 设置一个 Active Directory 和 Unix 账户

根据用户场景，执行以下步骤。

- 根据表 3. 用户场景的角色分配创建 AD 用户（如果你还没这么做）。
- 在 AD 中，创建以下 AD 组：
  - access\_mapredqueue\_development
  - access\_hbasetable\_phonerecords

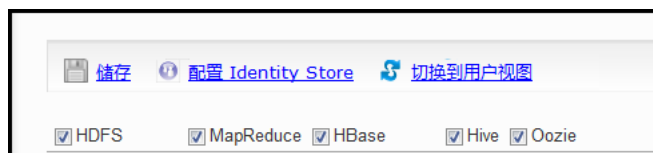


- access\_hivetable\_warehouse
  - hadoop-superusers
3. 在 AD 中，执行以下组任务：
    - 使 Chris Lee 成为 hadoop-superusers 组成员。
    - 使 George Smith 成为 access\_mapredqueue\_development 组成员。
    - 使 John Doe 成为 access\_hivetable\_warehouse 和 access\_hbasetable\_phonerecords 组成员。
  4. 在英特尔® Apache Hadoop\* 软件发行版集群中，确认节点对 HBase、Hive 和 MapReduce 有客户端访问。
  5. 在此节点上，确认以下有界面的 Unix 账户存在。如果不存在，请现在创建。
    - clee
    - gsmith
    - jdoe
  6. 确认每个 AD 用户能使用 AD 登录名作为 Unix 用户名登录到该节点。比如，Chris Lee 必须能够登录使用 Unix 用户名 *clee* 登录到 Linux 机器。

#### 2.8.4 启用基于角色的验证 Intel® Manager

要为 MapReduce、Hive 和 HBase 启用基于角色的验证，执行以下步骤。

1. 以管理员身份登录 Intel® Manager。
2. 在集群配置菜单，点击**安全**选项。
3. 在安全菜单，点击**权限管理**子页面。
4. 在权限管理子页面，选择以下选项：
  - MapReduce
  - HBase
  - Hive
5. 在权限管理子页面，点击**储存**链接。
6. 当出现二个对话框时，每次都点击**确定**。



7. 在 Intel® Manager 中，设置一个 identity store，使集群中的每个节点都能访问 AD 服务器。要了解详细信息，参见 2.5 设置和 Active Directory 服务器的通讯。
8. 要将这些设置部署到集群中的每个节点，执行章节 2.6 在集群中部署基于角色的验证变化中的步骤。

#### 2.8.5 为基于角色的验证配置 MapReduce 公平调度器

要为一个使用 Fair Scheduler 队列的 MapReduce 集群配置访问控制列表和基于角色的验证，执行以下步骤。

1. 以管理员身份登录 Intel® Manager。
2. 在集群概况菜单中，选择**控制面板**子页面。根据集群概述子页面信息，确认所有组件都已停止。
3. 在集群配置菜单中，点击**MapReduce**子页面。





4. 在 MapReduce 子页面，点击**简要配置**子页面。
5. 在任务分配模式栏，选择 **Fair** 选项。
6. 点击**配置 Fair Scheduler** 链接。

| 属性                                    | 值         |
|---------------------------------------|-----------|
| fairscheduler.userMaxJobsDefault      | 3         |
| mapred.fairscheduler.assignmultiple   | true      |
| mapred.fairscheduler.poolnameproperty | user.name |
| mapred.fairscheduler.preemption       | true      |
| mapred.fairscheduler.sizebasedweight  | true      |

7. 在 **mapred.fairscheduler.poolnameproperty** 下拉菜单中，选择 **mapred.job.queue.name** 选项。
8. 在简要配置页面中，点击**储存**链接。
9. 要将这些设置部署到集群中的每个节点，执行章节 2.6 在集群中部署基于角色的验证变化中的步骤。

## 2.8.6 在 Intel® Manager 中设置 hadoop-superusers 角色

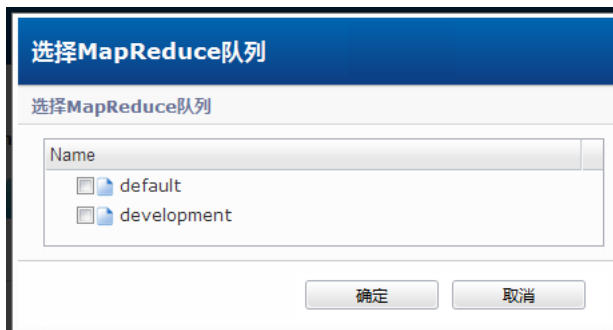
要创建表 2. 用户场景的角色中描述的 *hadoop-superusers* 角色，执行以下步骤：

1. 以管理员身份登录 Intel® Manager。
2. 确认以下服务已启动：
  - HDFS
  - ZooKeeper
  - MapReduce
  - HBase
  - Hive
3. 确认以下步骤已执行。
  - 2.5 设置和 Active Directory 服务器的通讯
  - 2.8.4 启用基于角色的验证 Intel® Manager
  - 2.8.5 为基于角色的验证配置 MapReduce 公平调度器
4. 在集群配置菜单，点击**安全**选项。
5. 在安全菜单，点击**权限管理**子页面。
6. 在角色视图中，点击**添加**。
7. 在**名称**栏内，输入 **hadoop-superusers**。

8. 在**描述**栏内，输入 `administer hdfs, mapred, hbase, and hive`。



9. 在 HDFS 子页面，选择**管理 HDFS，成为 DFSAdmin** 选项。
10. 执行以下步骤指定 MapReduce 权限。
- 选择 **MapReduce** 子页面。
  - 在 MapReduce 子页面中，选择**提交 Job 到指定的队列**选项。
  - 点击**提交 Job 到指定的队列**选项旁的**点击这儿添加**链接。



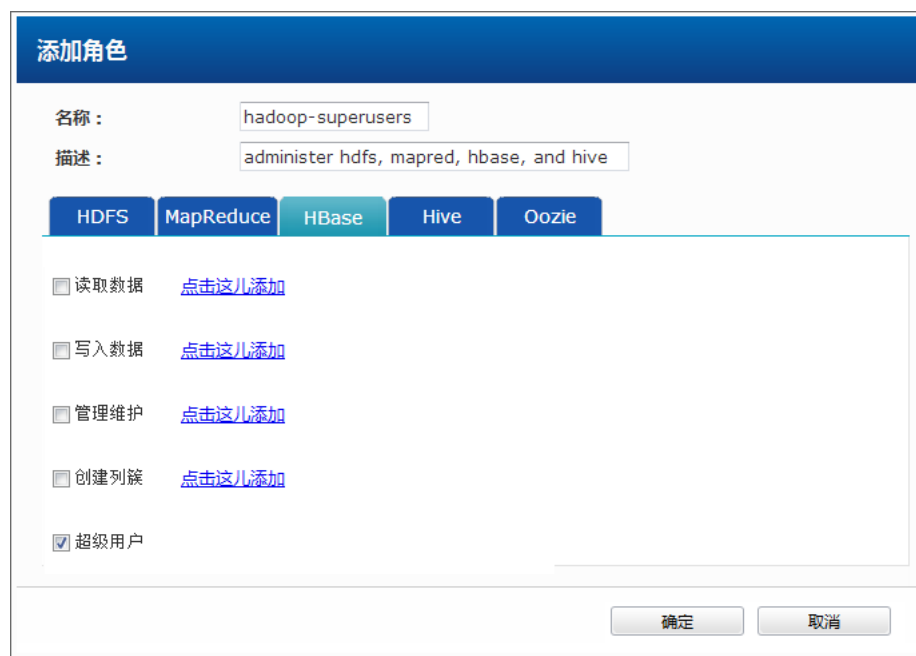
- 在选择 MapReduce 队列对话框，选择 **development** 选项，然后点击**确定**。
- 在 MapReduce 子页面中，选择**管理指定队列中的 Job** 选项。
- 点击**管理指定队列中的 Job** 选项旁的**点击这儿添加**链接。
- 在选择 MapReduce 队列对话框，选择 **development** 选项，然后点击**确定**。

- h. 选择**管理 MapReduce**，成为 MRAdmin 选项。



11. 执行以下步骤指定 HBase 权限。

- a. 选择 **HBase** 子页面。  
b. 选择**超级用户**选项。



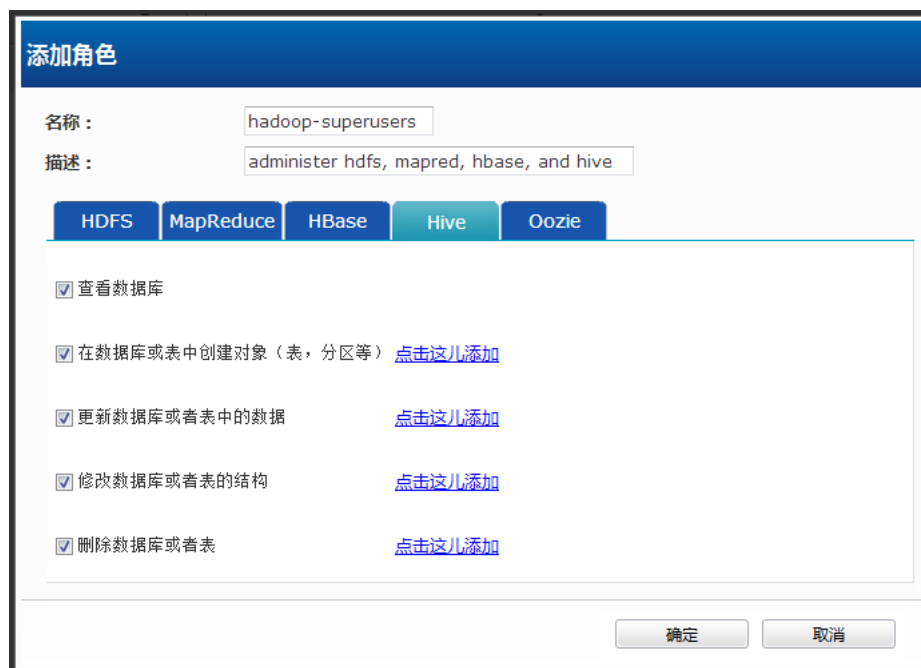
12. 执行以下步骤指定 Hive 权限。

- a. 点击 **Hive** 子页面。  
b. 选择**查看数据库**选项。  
c. 选择**在数据库或表中创建对象**选项。

- d. 点击在数据库或表中创建对象选项旁的[点击这儿添加](#)链接。



- e. 在选择 Hive databases, tables 对话框, 选择 **default** 选项, 然后点击**确定**。
- f. 选择**更新数据库或表中的数据**选项。
- g. 点击**更新数据库或表中的数据**选项旁的[点击这儿添加](#)链接。
- h. 在选择 Hive databases, tables 对话框, 选择 **default** 选项, 然后点击**确定**。
- i. 选择**修改数据库或者表中的结构**选项。
- j. 点击**更新数据库或表中的数据**选项旁的[点击这儿添加](#)链接。
- k. 在选择 Hive databases, tables 对话框, 选择 **default** 选项, 然后点击**确定**。
- l. 选择**删除数据库或者表**选项。
- m. 点击**删除数据库或者表**选项旁的[点击这儿添加](#)链接。
- n. 在选择 Hive databases, tables 对话框, 选择 **default** 选项, 然后点击**确定**。



13. 在添加角色对话框, 点击**确定**按钮。
14. 在角色视图区域, 点击 *hadoop-superusers* 角色的用户 / 组链接。



15. 在分配用户和组对话框，执行以下步骤。
  - a. 在搜索栏内，输入 **superusers**，然后按下 Enter。
  - b. 选择 **hadoop-superusers** 选项。

| 类别                                  | 名字                |
|-------------------------------------|-------------------|
| <input checked="" type="checkbox"/> | hadoop-superusers |

- c. 点击**确定**。
16. 在权限管理子页面，点击**储存**链接。

| 授权角色                                                  | 描述                     | 组件                        | 用户/组                |
|-------------------------------------------------------|------------------------|---------------------------|---------------------|
| <input checked="" type="checkbox"/> hadoop-superusers | administer hdfs, mapre | HDFS MAPREDUCE HBASE HIVE | [hadoop-superusers] |

17. 要将这些设置部署到集群中的每个节点，执行章节 2.6 在集群中部署基于角色的验证变化中的步骤。

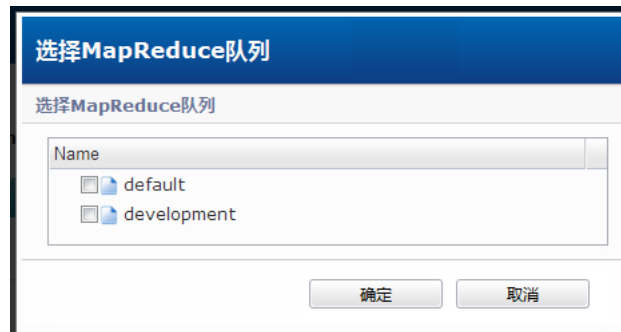
## 2.8.7 在 Intel® Manager 中设置 access\_mapredqueue\_development 角色

要创建表 2. 用户场景的角色中描述的 *access\_mapredqueue\_development* 角色，执行以下步骤：

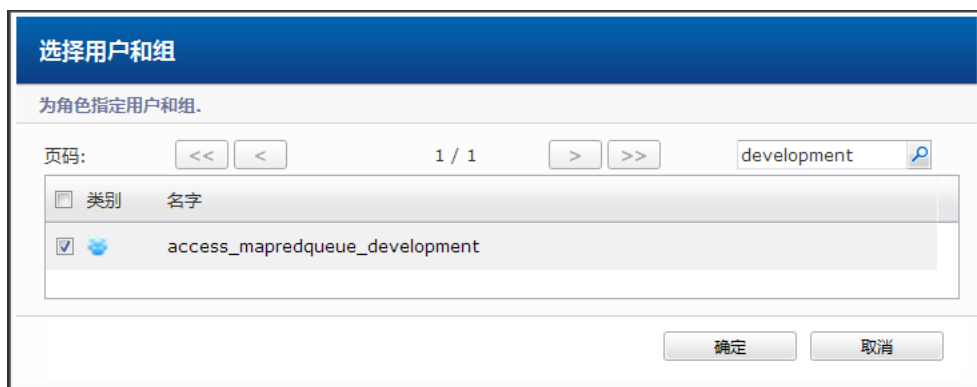
1. 以管理员身份登录 Intel® Manager。
2. 确认以下服务已启动：
  - HDFS
  - ZooKeeper
  - MapReduce
  - HBase
  - Hive
3. 确认以下步骤已执行。
  - [2.5 设置和 Active Directory 服务器的通讯](#)
  - [2.8.4 启用基于角色的验证 Intel® Manager](#)

- 2.8.5 为基于角色的验证配置 MapReduce 公平调度器

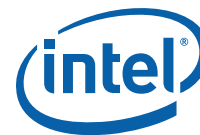
- 在集群配置菜单，点击**安全**选项。
- 在安全菜单，点击**权限管理**子页面。
- 在角色视图中，点击**添加**。
- 在**名称**栏内，输入 `access_mapredqueue_development`。
- 在**描述**栏内，输入 `submit jobs to mapred development queue`。
- 选择 **MapReduce** 子页面。
- 在 MapReduce 子页面中，选择**提交 Job 到指定的队列**选项。
- 点击**提交 Job 到指定的队列**选项旁的**点击这儿添加链接**。



- 在选择 MapReduce 队列对话框，选择 `development` 选项，然后点击**确定**。
- 在添加角色对话框，点击**确定**按钮。
- 在角色视图区域，点击 `access_mapredqueue_development` 角色的用户 / 组链接。
- 在分配用户和组对话框，执行以下步骤。
  - 在搜索栏内，输入 `development`，然后按下 Enter。
  - 选择 `access_mapredqueue_development` 选项。



- 点击**确定**。
- 在权限管理子页面，点击**储存**链接。
  - 要将这些设置部署到集群中的每个节点，执行章节 2.6 在集群中部署基于角色的验证变化中的步骤。



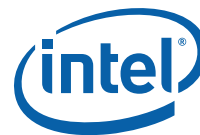
## 2.8.8 在 Intel® Manager 中设置 access\_hbasetable\_phonerecords 角色

要创建表 2. 用户场景的角色中描述的 `access_hbasetable_phonerecords` 角色，执行以下步骤：

1. 以管理员身份登录 Intel® Manager。
2. 确认以下服务已启动：
  - HDFS
  - ZooKeeper
  - MapReduce
  - HBase
  - Hive
3. 确认以下步骤已执行。
  - [2.5 设置和 Active Directory 服务器的通讯](#)
  - [2.8.4 启用基于角色的验证 Intel® Manager](#)
4. 在集群配置菜单，点击**安全**选项。
5. 在安全菜单，点击**权限管理**子页面。
6. 在角色视图中，点击**添加**。
7. 在**名称**栏内，输入 `access_hbasetable_phonerecords`。
8. 在**描述**栏内，输入 `access hbase phone records table`。
9. 选择 **HBase** 子页面。
10. 选择**读取数据**选项。
11. 点击**读取数据**选项旁的**点击这儿添加**链接。



12. 在选择 HBase databases, tables 对话框，选择 `phone_records` 选项，然后点击**确定**。
13. 选择**写入数据**选项。
14. 点击**写入数据**选项旁的**点击这儿添加**链接。



15. 在选择 HBase databases, tables 对话框, 选择 **phone\_records** 选项, 然后点击**确定**。

16. 在添加角色对话框, 点击**确定**按钮。
17. 在角色视图区域, 点击 *access\_hbasetable\_phonerecords* 角色的用户 / 组链接。
18. 在分配用户和组对话框, 执行以下步骤。
- 在搜索栏内, 输入 **phonerecords**, 然后按下 Enter。
  - 选择 **access\_hbasetable\_phonerecords** 选项。
  - 点击**确定**。
19. 在权限管理子页面, 点击**储存**链接。
20. 要将这些设置部署到集群中的每个节点, 执行章节 2.6 在集群中部署基于角色的验证变化中的步骤。

## 2.8.9 在 Intel® Manager 中设置 access\_hivetable\_warehouse 角色

要创建表 2. 用户场景的角色中描述的 *access\_hivetable\_warehouse* 角色, 执行以下步骤:

- 以管理员身份登录 Intel® Manager。
- 确认以下服务已启动:
  - HDFS
  - ZooKeeper
  - MapReduce
  - HBase
  - Hive
- 确认以下步骤已执行。
  - [2.5 设置和 Active Directory 服务器的通讯](#)
  - [2.8.4 启用基于角色的验证 Intel® Manager](#)



4. 在集群配置菜单，点击**安全**选项。
5. 在安全菜单，点击**权限管理**子页面。
6. 在角色视图中，点击**添加**。
7. 在**名称**栏内，输入 `access_hivetable_warehouse`。
8. 在**描述**栏内，输入 `access hive warehouse table`。
9. 点击 **Hive** 子页面。
10. 选择**在数据库或表中创建对象**选项。
11. 点击**在数据库或表中创建对象**选项旁的**点击这儿添加**链接。



12. 在选择 Hive databases, tables 对话框，点击 **warehouse** 选项，然后点击**确定**。
13. 选择**更新数据库或表中的数据**选项。
14. 点击**更新数据库或表中的数据**选项旁的**点击这儿添加**链接。
15. 在选择 Hive databases, tables 对话框，点击 **warehouse** 选项，然后点击**确定**。





16. 在添加角色对话框，点击**确定**按钮。
17. 在角色视图区域，点击 `access_hivetable_warehouse` 角色的用户 / 组链接。
18. 在分配用户和组对话框，执行以下步骤。
  - a. 在搜索栏内，输入 **warehouse**，然后按下 Enter。
  - b. 选择 **access\_hivetable\_warehouse** 选项。
  - c. 点击**确定**。
19. 在权限管理子页面，点击**储存**链接。

| <input type="checkbox"/> 授权角色                                        | 描述                            | 组件                                        | 用户/组                                             |
|----------------------------------------------------------------------|-------------------------------|-------------------------------------------|--------------------------------------------------|
| <input type="checkbox"/> <code>hadoop-superusers</code>              | administer hdfs, mapred, hbas | <a href="#">HDFS MAPREDUCE HBASE HIVE</a> | <a href="#">[hadoop-superusers]</a>              |
| <input type="checkbox"/> <code>access_mapredqueue_development</code> | submit jobs to mapred develop | <a href="#">MAPREDUCE</a>                 | <a href="#">[access_mapredqueue_development]</a> |
| <input type="checkbox"/> <code>access_hbasetable_phonerecords</code> | access hbase phone records t  | <a href="#">HBASE</a>                     | <a href="#">[access_hbasetable_phonerecords]</a> |
| <input type="checkbox"/> <code>access_hivetable_warehouse</code>     | access hive warehouse table   | <a href="#">HIVE</a>                      | <a href="#">[access_hivetable_warehouse]</a>     |

20. 要将这些设置部署到集群中的每个节点，执行章节 2.6 在集群中部署基于角色的验证变化中的步骤。



## 3.0 使用应用层加密实现数据保密

---

能被某些人或某些程序读取或处理的数据，也称为明文（plaintext）。默认情况下，Apache Hadoop\* 集群中处理和存储的数据为明文。由于潜在的数据敏感性，比如，个人身份信息（PII），数据保密要求不允许将此类数据以明文存储。

数据保密是确保只有具备权限的一方才能访问明文数据的一个过程。加密是实施数据保密的其中一个方法，加密是将数据编码、以使得仅被授权方可读取数据的一个过程。

英特尔® Apache Hadoop\* 软件发行版支持通过 Java API 的应用层加密。你可以使用这些 API 来设计应用程序，以加密或解密 HDFS 中存储的数据。以下章节解释了这一工作原理，以及如何设计 Java 应用程序来执行此类加密。

### 3.1 加密的工作原理

在一个加密表中，通常应用加密算法来将数据从明文转换为密文（ciphertext）。密文是一种形式上被打乱的数据，没有程序或人员可以读懂。解密是加密的相反过程，解密使用同样的加密算法来将数据从密文重新转换为原来的明文。

解密和加密算法都需要一个密钥。密钥是一个数值，是加密算法中的一个输入参数，用于加密或解密数据。密钥可确保算法的结果是唯一的而且是保密的。没有正确的密钥，你不可以成功加密或解密数据。因此，数据保密性和你是否能访问解密密文的密钥有关。通过仅授权密钥的访问给那些有权查看数据的程序和人员，你可以确保只有被授权方可以明文形式读取数据。

### 3.2 支持的加密方式

英特尔® 发行版提供的 Java API 支持非对称和对称加密。

#### 3.2.1 对称加密

要提供点到点的数据加密，对称加密支持以下方式：

- 通过 FileSystem API 访问 HDFS  
Java 程序能使用 FileSystem API 来加密流向 HDFS 的数据，并解密从 HDFS 流出的数据。也就是说，数据可加密后存入 HDFS，并从 HDFS 取出后进行解密。使用文件名来表示，你可以指定用怎样的密钥来加密一个或一组文件。
- 通过 MapReduce 访问 HDFS  
Java 程序可使用 MapReduce API 从 HDFS 获得加密数据，在 MapReduce 作业中处理数据，并将作业的输出结果加密，然后流向 HDFS。API 为开发者提供了是否加密中间结果、最终结果，或者二者都加密的灵活选择。使用文件名来表示，你可以指定用怎样的密钥来加密一个或一组文件。

对称加密使用 AES 算法，密钥长度为 128 位或 256 位。

#### 3.2.2 非对称加密

MapReduce 应用程序提交给 JobTracker 的作业配置文件有可能包含重要的安全参数。要保护数据不被恶意用户通过网络通讯偷听，英特尔® 发行版支持对作业配置文件的点到点非对称加密。



在 MapReduce 应用程序提交作业到 JobTracker 之前，应用程序可使用公共密钥来加密作业配置文件。当 JobTracker 和 TaskTracker 接收到作业配置文件，每个 daemon 使用相应的 RSA 私钥来解密作业配置文件。

### 3.3 密钥是如何管理的？

所有密钥和证书都存储在 Java KeyStore 中。有些公共密钥和相关联的 X.509 证书可能存储在 Java TrustStore 中。当一个应用程序被设计用来执行加密时，应用程序必须能访问本地文件系统的 KeyStore，你需要知道用于加密的密钥的 alias，而且应用程序必须有访问 KeyStore 和私钥的密码。具备这些信息后，应用程序可从 KeyStore 获取密钥，并使用该密钥加密或解密数据。

### 3.4 需要加密的 API 和 JavaDoc 存在哪里？

当英特尔® Apache Hadoop\* 软件发行版被安装在某个系统后，以下文件将被创建 `/usr/lib/hadoop/hadoop-core-1.0.3-Intel.jar`。 `hadoop-core-1.0.3-Intel.jar` 包含了用于加密 Apache Hadoop\* 集群中数据的 API。解释如何使用 API 的 Javadoc 文档位于 `/usr/share/doc/hadoop-1.0.3-Intel/api`。

### 3.5 编辑执行加密的 Java 程序

要编译使用 API 加密 HDFS 的 Java 程序，以下文件必须已存放在类的路径：

- `/usr/lib/hadoop/hadoop-core-1.0.3-Intel.jar`
- `/usr/lib/hadoop/lib/commons-cli-1.2.jar`

### 3.6 先决条件

要设计一个为 Apache Hadoop\* 集群中的文件加密的 Java 应用程序，你必须符合以下要求：

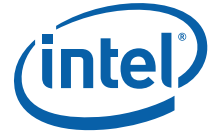
#### 3.6.1 总体要求

- 对 Java KeyStores 和 TrustStore 有一个基本了解，包括创建非对称密钥对、创建对称密钥、列出 KeyStore 的可用密钥和证书，以及从 KeyStore 提取密钥和证书。
- 所有的密钥和证书必须用 Java keytool 来生成。
- 加密和解密通过 OpenSSL 来执行。确认集群中所有节点上已安装 OpenSSL 1.0.0 或更高版本。
- 为使加密操作达到最好的性能，推荐使用 OpenSSL 1.0.1c，且集群中的所有节点使用 AES-NI 技术。
- HDFS 上加密的文件名必须有前缀 `.intel_aes`。
- KeyStore 的文件权限必须为 600 或 640。
- 集群中使用的 Java Runtime Environment 必须启用无限制的 JCE。

#### 3.6.2 非对称加密的要求

对于作业配置文件的非对称加密，必须符合以下要求：

- 必须使用 RSA 加密算法。
- 私钥大小必须至少为 1024 位。
- 非对称密钥对必须储存在 KeyStore，且 KeyStore 的类型必须是 JKS。
- 对于解密作业配置文件的 JobTracker 和 TaskTracker，必须符合以下要求：
  - KeyStore 的文件许可必须为 640。
  - KeyStore 的文件所有者必须为 `root`。



- KeyStore 的组名必须是 *hadoop*。
- 在集群的每个节点上，KeyStore 必须存放在以下路径：*/usr/lib/hadoop/keystore*。
- 公共密钥必须存放在 TrustStore。
- 要在作业提交到 JobTracker 前加密配置文件，MapReduce 应用程序必须可访问 TrustStore 的公共密钥。这意味着 TrustStore 必须在应用程序执行的机器上，且运行应用程序的用户必须对 TrustStore 有读的权限。
- 还必须有个符合章节 3.6.4 KeyStore 密码文件的要求要求的密码。

### 3.6.3 对称加密的要求

对于对称加密，必须符合以下要求：

- 必须使用 AES 加密算法。
- 密钥大小必须为 128 位或 256 位。
- 储存密钥的 KeyStore 类型必须为 JCEKS。
- 对于运行 Java 应用程序来执行加密的用户，KeyStore 必须存放在 Unix 根目录，或者存放在 */usr/lib/hadoop/keystore*。
- 应用程序开发者必须知道 KeyStore 的密码，以及 KeyStore 中用于加密的密钥。
- 应用程序开发者必须知道 KeyStore 的位置及其文件。
- 应用程序开发者必须知道要用来加密或解密数据的密钥的别名。
- 执行加密的 Java 应用程序必须在存储 KeyStore 和密码文件的系统上运行。应用程序必须对 KeyStore 和密码文件有读的权限。
- 还必须有个符合章节 3.6.4 KeyStore 密码文件的要求要求的密码。

### 3.6.4 KeyStore 密码文件的要求

要从 KeyStore 获得密钥，服务或应用程序必须有 KeyStore 密码和需要从 KeyStore 获得的密钥的密码。要让应用程序或服务可访问密码，你需要创建一个文本文件用于存储密码明文。然后，当应用程序需要访问 KeyStore 时，它能从文本文件中读取密码。

要从文件中获得密码并正确使用密码，密码文件必须为以下格式：

```
KeyStore.password={keystore password}
{keyalias}.password={key password}
{keyalias}.password={key password}
```

要了解文件格式，参考以下情形：

- KeyStore 密码是 *123456*。
- KeyStore 中有二个密钥。
- 第一个密钥的别名是 *key001*，其密码是 *pa\$\$wd123*。
- 第二个密钥的别名是 *key002*，其密码是 *usingreallystrongencryption*。

在以上情形中，密码文件的格式如下：

```
KeyStore.password=123456
key001.password=pa$$wd123
key002.password=usingreallystrongencryption
```



在加密机制中保护文件的机密性至关重要。如果有人或程序有 KeyStore 和密码文件的读取权限，则他们可解密通过 KeyStore 中的密钥加密的任何数据。因此，你必须对 KeyStore 密码文件采取以下安全预警措施：

- 对于对称加密，密码文件必须符合以下要求：
  - 对于运行 MapReduce 或 HDFS 应用程序的用户，密码文件必须存储在 UNIX 根目录。
  - 文件权限必须为 400。
  - 文件所有者必须是于运行 MapReduce 或 HDFS 应用程序的 UNIX 用户。
  - 文件名的前缀必须是 *.passwords.*。
- 对于非对称加密，KeyStore 密码文件必须符合以下要求：
  - 文件权限必须为 440。
  - 文件所有者必须是 *root*。
  - 文件的组名必须是 *hadoop*。
  - 在集群的每个节点上，密码文件必须存放在以下路径： */usr/lib/hadoop/keystore*。
  - 文件名的前缀必须是 *.passwords.*。

## 3.7 创建 Java KeyStore

### 3.7.1 创建对称加密的 Java KeyStore

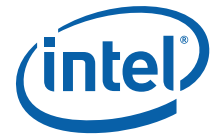
要创建对称加密的 Java KeyStore，执行以下步骤：

1. 登录到管理节点。
2. 使用 `cd` 命令进入根目录。
3. 在根目录下，创建一个名为 *keystore* 的目录。
4. 使用 `cd` 命令进入 *keystore* 目录。
5. 要在新的 KeyStore 中创建密钥，你需要决定以下事项：
  - 密钥的别名
  - 密钥的密码
  - 密钥的大小密钥大小可以是 128 位或 256 位。
  - KeyStore 的密码
  - KeyStore 的文件名
6. 要在新的 KeyStore 中创建密钥，执行以下命令：

```
keytool -genseckey -alias {keyalias} -keypass {keypassword} -storepass {keystore password} -keyalg AES -keysize {key size in bits} -KeyStore {filename of keystore} -storetype JCEKS
```

参考以下情形：

- 密钥的别名是 *key01*。
- 密钥的密码是 *pa\$\$wd123*。
- 密钥的大小是 128 位。
- KeyStore 密码是 *123456*。
- KeyStore 文件名是 *idhencryption.keystore*。



在以上情形中，你需要执行以下命令来创建密钥和 KeyStore：

```
keytool -genseckey -alias key01 -keypass pa$$wd123 -storepass 123456 -
keyalg AES -keysize 128 -KeyStore idhencryption.keystore -storetype
JCEKS
```

7. 一旦 KeyStore 被创建，将文件权限更改为 600。
8. 在 *keystore* 目录中，创建一个名为 *symmetrickeystore.passwords* 的空白文本文件。
9. 一旦 *symmetrickeystore.passwords* 被创建，将文件权限更改为 400。
10. 在 *symmetrickeystore.passwords* 中，输入以下内容：

```
KeyStore.password={keystore password}
{keyalias}.password={key password}
```

11. 在 *symmetrickeystore.passwords* 中，执行以下步骤。
  - a. 将字符串 *{keystore password}* 替换为 KeyStore 的密码。
  - b. 将字符串 *{keyalias}* 替换为你在 KeyStore 中创建的密钥别名。
  - c. 将字符串 *{key password}* 替换为你在 KeyStore 中创建的密钥密码。
12. 参考以下情形：
  - 密钥的别名是 *key01*。
  - 密钥的密码是 *pa\$\$wd123*。
  - KeyStore 密码是 *123456*。
 在以上情形中，这就是文件 *keystore.passwords* 中的内容。

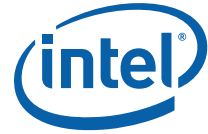
```
KeyStore.password=123456
key001.password=pa$$wd123
```

13. 在设计一个 MapReduce 应用程序时，当从它从 HDFS 获取数据作为输入文件时，你可以用一个对称密钥来解密数据；在 MapReduce 作业将输出结果写入到 HDFS 时，用一个不同的对称密钥来加密。如果你要这么做，你需要至少二个 Java KeyStore 中的对称密钥。

### 3.7.2 创建非对称加密的 Java KeyStore 和 TrustStore。

要创建对称加密的 Java KeyStore 和 TrustStore，执行以下步骤：

1. 登录到管理节点。
2. 成为 root 用户（如果你还没这么做）。
3. 如果目录 */usr/lib/hadoop/keystore* 不存在，请现在创建它。
4. 将 */usr/lib/hadoop/keystore* 的文件权限设置为 744。
5. 通过 *cd* 命令进入 */usr/lib/hadoop/keystore*。
6. 要在新的 KeyStore 中创建密钥对，你需要决定以下事项：
  - 密钥的别名
  - 密钥的密码
  - 密钥的大小密钥大小可以是 1024 位或更高。
  - KeyStore 的密码



- KeyStore 的文件名
- X.509 证书的标识名 (DN)

7. 要在新的 KeyStore 中创建密钥对, 执行以下命令:

```
keytool -genkey -alias {keyalias} -keyalg RSA -keystore {keystore
filename} -storepass {keystore password} -keypass {key password} -
dname "{DN for X.509 certificate}" -storetype JKS -keysize {size of
key}
```

8. 参考以下情形:

- 密钥的别名是 *CLUSTERPRIVATEKEY*。
- 密钥的密码是 *pa\$\$wd123*。
- 密钥的大小是 1024 位。
- KeyStore 密码是 *123456*。
- KeyStore 文件名是 *clusterprivate.keystore*。
- X.509 证书的标识名是 *CN=John Doe OU=Development, O=Intel, L=Chicago, S=IL, C=US*。

在以上情形中, 你需要执行以下命令来创建密钥对和 KeyStore:

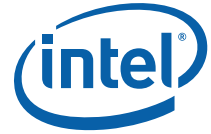
```
keytool -genkey -alias CLUSTERPRIVATEKEY -keyalg RSA -keystore
clusterprivate.keystore -storepass 123456 -keypass pa$$wd123 -dname
"CN=John Doe OU=Development, O=Intel, L=Chicago, S=IL, C=US" -
storetype JKS -keysize 1024
```

- 一旦 KeyStore 被创建, 将文件权限更改为 640。
- 将 KeyStore 文件所有者更改为 *root*。
- 将 KeyStore 文件组更改为 *hadoop*。
- 在 *keystore* 目录中, 创建一个名为 *asymmetrickeystore.passwords* 的空白文本文件。
- 一旦 *asymmetrickeystore.passwords* 被创建, 将文件权限更改为 440。
- 在 *asymmetrickeystore.passwords* 中, 输入以下内容:

```
KeyStore.password={keystore password}
{keyalias}.password={key password}
```

- 在 *asymmetrickeystore.passwords* 中, 执行以下步骤。
  - 将字符串 *{keystore password}* 替换为 KeyStore 的密码。
  - 将字符串 *{keyalias}* 替换为你在 KeyStore 中创建的密钥别名。
  - 将字符串 *{key password}* 替换为你在 KeyStore 中创建的密钥密码。
- 将 *asymmetrickeystore.passwords* 文件所有者更改为 *root*。
- 将 *asymmetrickeystore.passwords* 组改为 *hadoop*。
- 参考以下情形:
  - 密钥的别名是 *CLUSTERPRIVATEKEY*。
  - 密钥的密码是 *pa\$\$wd123*。
  - KeyStore 密码是 *123456*。





在以上情形中，这就是文件 *keystore.passwords* 中的内容。

```
KeyStore.password=123456
CLUSTERPRIVATEKEY.password=pa$$wd123
```

19. 执行以下步骤创建 TrustStore:

- a. 使用 `keytool` 从 KeyStore 中提取公共密钥，并保存到名为 *public.cert* 的文件中。比如，如果私钥的别名是 *CLUSTERPRIVATEKEY*，且 KeyStore 的文件名是 *clusterprivate.keystore*，则执行以下命令：

```
keytool -export -alias CLUSTERPRIVATEKEY -keystore
clusterprivate.keystore -rfc -file public.cert -storepass {truststore
password}
```

- b. 使用 `keytool` 命令来创建包含有从 KeyStore 提取的公共证书的 TrustStore。比如，如果公共密钥的别名是 *CLUSTERPUBLICKEY*，且 TrustStore 的文件名是 *clusterpublic.trustore*，则执行以下命令：

```
keytool -import -alias CLUSTERPUBLICKEY -file public.cert -keystore
clusterpublic.truststore -storepass 123456
```

20. 一旦 TrustStore 被创建，将文件权限更改为 644。
21. 将 TrustStore 文件所有者更改为 *root*。
22. 将 TrustStore 文件组更改为 *hadoop*。
23. 在 *keystore* 目录中，创建一个名为 *truststore.passwords* 的空白文本文件。
24. 一旦 *truststore.passwords* 被创建，将文件权限更改为 644。
25. 在 *truststore.passwords* 中，输入以下内容：

```
KeyStore.password={trustore password}
```

26. 在 *truststore.passwords* 文件中，将字符串 *{truststore password}* 替换为 TrustStore 的密码。
27. 将 */usr/lib/hadoop/keystore* 目录及其所属文件拷贝到集群中的所有其他节点。在每个节点上，*keystore* 目录必须在 */usr/lib/hadoop/* 目录下。



### 3.8 创建加密和解密数据的 HDFS 应用程序

在目录 A.1 HDFS 应用程序执行加密中，有一个示例程序，演示了将文件存放到 HDFS 如何加密，以及从 HDFS 下载文件如何解密。要掌握如何设计此类程序，以下步骤将通过示例程序的关键部分来指导你如何让应用程序加密和解密数据。

1. 参考以下情形：

```
boolean encrypt = "-e".equals(args[0]);
String keyStoreFile = args[1];
String keyStorePasswordFile = args[2];
String keyAlias = args[3];
String inputFile = args[4];
String outputFile = args[5];
```

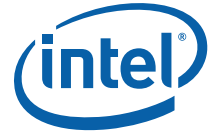
定义能将命令行传递到 Java 程序的参数列表参数定义参见以下列表：

- a. encrypt — 注明文件是否被加密的 boolean 值。如果是 true，则输入文件是位于本地文件系统的文件，它在输入 HDFS 时被加密。如果是 false，则输入文件是位于 HDFS 的一个加密文件，它在存入本地文件系统时被解密。
- b. keyStoreFile — 注明本地文件系统上 KeyStore 文件的位置。值必须遵守文件 URI 句法。比如，如果文件位于 `/tmp/test.keystore`，则值为 `file:///tmp/test.keystore`。
- c. keyStorePasswordFile — 注明本地文件系统上 KeyStore 密码文件的位置。值必须遵守文件 URI 句法。比如，如果文件位于 `/tmp/test.passwords`，则值为 `file:///tmp/test.passwords`。
- d. keyAlias — 可从 KeyStore 中获取的对称密钥别名
- e. inputFile — 参数含义根据加密参数值的不同而变化
  - i. 如果加密值为 true，则注明流向 HDFS 要被加密的本地文件系统上的文件位置。值必须遵守文件 URI 句法。比如，如果文件位于 `/tmp/test.keystore`，则值为 `file:///tmp/test.keystore`。
  - ii. 如果加密值为 false，则注明 HDFS 上加密文件的位置。
- f. outputFile — 参数含义根据加密参数值的不同而变化
  - i. 如果加密值为 false，则 HDFS 上的加密文件将被解密。这一参数注明了明文文件被保存到本地文件系统的位置。值必须遵守文件 URI 句法。比如，如果文件位于 `/tmp/test.keystore`，则值为 `file:///tmp/test.keystore`。
  - ii. 如果加密值为 true，则注明 HDFS 上加密文件的位置。

2. 参考以下情形：

```
Configuration conf = new Configuration();
KeyStoreKeyProvider.KeyStoreConfig keyStoreParameters = new
KeyStoreKeyProvider.KeyStoreConfig(keyStoreFile, null,
keyStorePasswordFile, "JCEKS", false);
KeyStoreKeyProvider keyProvider = new KeyStoreKeyProvider();
keyProvider.setConf(conf);
keyProvider.init(keyStoreParameters.toBytes());
```

KeyStoreKeyProvider 类能从 JCEKS KeyStore 获取对称密钥。KeyStoreConfig 类注明 KeyStoreKeyProvider 类获取对称密钥需要的参数。在以上示例中，KeyStoreConfig 类通过使用以下参数调用构造函数来实例化。



- a. `keyStoreUrl` — 注明本地文件系统上 KeyStore 文件的位置。在以上示例中，这由 `keyStoreFile` 变量定义。值必须遵守文件 URI 句法。比如，如果文件位于 `/tmp/test.keystore`，则值为 `file:///tmp/test.keystore`。
- b. `keyStorePassword` — 注明访问 KeyStore 的密码由于已使用密码文件，这一字段的值为 `null`。
- c. `keyStorePasswordFile` — 注明本地文件系统上 KeyStore 密码文件的位置。在以上示例中，这由 `keyStorePasswordFile` 变量定义。值必须遵守文件 URI 句法。比如，如果文件位于 `/tmp/test.passwords`，则值为 `file:///tmp/test.passwords`。
- d. `keyStoreType` — 注明使用的 keystore 类型。可能的值为 JCEKS 或 JKS。在以上示例中，因为执行的是对称加密，类型必须为 JCEKS。
- e. `sharedPassword` — `boolean` 值。如果密钥和 keystore 的密码相同，则值为 `true`。如果密钥和 keystore 的密码不同，则值为 `false`。在以上示例中，值为 `false`。

当新的 `KeyStoreKeyProvider` 对象创建后，类包含打包在 `KeyStoreConfig` 类中的配置参数。`setConf` 方法将 `KeyStoreKeyProvider` 类的对象放入发送到 `PrimaryNameNode` 的 XML 配置文件。

### 3. 参考以下情形：

```
Key[] keys = keyProvider.getKeys(new String[] { keyAlias });
```

`getKeys` 方法使用 `KeyProvider` 对象中的信息从 Java KeyStore 中获得一个或多个密钥，并将这些密钥存储在阵列中。这一方法仅能获得有别名的密钥。如果密钥存在于 KeyStore 但别名没有传递给该方法，则密钥不能从 KeyStore 获得。在以上示例中，考虑以下情形：

- 只有一个别名可被传递给方法，因此你只能从 KeyStore 获得能解析此别名的单个密钥。
- 别名是在 Java 应用程序执行时通过命令行参数传递的。

### 4. 参考以下情形：

```
CryptoContext cryptoContext = new CryptoContext();
cryptoContext.setKey(keys[0]);
keys[0].setCryptographicLength(128);
AESCodec codec = new AESCodec();
codec.setCryptoContext(cryptoContext);
((AESCodec) codec).setConf(conf);
```

类和方法描述如下：

- `CryptoContext` 类注明了哪些密钥用来执行加密和解密。
- `setKey` 方法注明了哪些通过 `KeyProvider` 类中的 `getKeys` 方法创建的阵列中的密钥被使用。
- `setCryptographicLength` 注明了从阵列中获得的密钥大小。
- `AESCodec` 类注明用来执行对称加密的 AES。
- `setCryptoContext` 方法注明了 AES 解码器使用 `CryptoContext` 类中定义的对称密钥来执行加密和解密。
- `setConf` 方法将 `AESCode` 类的对象放入发送到 `PrimaryNameNode` 的 XML 配置文件。



### 3.9 创建加密和解密数据的 MapReduce 应用程序

在附录 A.2 [MapReduce 应用程序执行加密](#) 中，有一个 MapReduce 应用程序的示例，它从 HDFS 获得加密文件，使用对称密钥解密文件，在某个作业中使用这些文件的数据，然后使用对称密钥加密 reducer 的输出结果并写入到 HDFS。这些应用程序假设 MapReduce 在从 HDFS 获得数据输入前，HDFS 的数据已使用 HDFS 应用程序（比如附录 A.1 [HDFS 应用程序执行加密](#) 中的这一示例）加密。

要掌握如何设计此类 MapReduce 程序，以下步骤将通过示例程序的关键部分来指导你如何让应用程序加密和解密数据。

1. 参考以下情形：

```
String asymKSFilePub = args[0];
String asymKSPassFilePub = args[1];
String asymKeyPvtFileName = args[2];
String asymPvtKeyAlias = args[3];
String asymPubKeyAlias = args[4];
String symKSFile = args[5];
String symKSPassFile = args[6];
String inputSymKeyAlias = args[7];
String outputSymKeyAlias = args[8];
String inputFile = args[9];
String outputFile = args[10];
```

定义能将命令行传递到 Java 程序的参数列表参数定义参见以下列表：

- a. `asymKSFilePub` — 包含用于加密作业配置文件的公共密钥的 TrustStore 文件名。TrustStore 必须存放在以下路径：`/usr/lib/hadoop/keystore`。
- b. `asymKSPassFilePub` — 本地文件系统的 TrustStore 密码文件名。密码文件必须存放在以下路径：`/usr/lib/hadoop/keystore`。
- c. `asymKeyPvtFileName` — 包含解密作业配置文件的私钥的非对称 KeyStore 文件名。KeyStore 必须存放在以下路径：`/usr/lib/hadoop/keystore`。  
KeyStore 密码文件的文件名必须和 KeyStore 文件名一致，并带有前缀 `.passwords`。比如，如果 KeyStore 文件名是 `asymmetricencrypt.keystore`。则密码文件名是 `asymmetricencrypt.keystore.passwords`。
- d. `asymPvtKeyAlias` — 从 KeyStore 获取的、用于解密作业配置文件的 RSA 密钥别名。
- e. `asymPvtKeyAlias` — 从 TrustStore 获取的、用于加密作业配置文件的公共密钥别名。
- f. `symKSFile` — 注明本地文件系统中对称 KeyStore 文件的位置。文件包含用于加密和解密 HDFS 数据的对称密钥。值必须遵守文件 URI 句法。比如，如果文件位于 `/tmp/test.keystore`，则值为 `file:///tmp/test.keystore`。
- g. `symKSPassFile` — 注明本地文件系统中对称 KeyStore 密码文件的位置。值必须遵守文件 URI 句法。比如，如果文件位于 `/tmp/test.passwords`，则值为 `file:///tmp/test.passwords`。
- h. `inputSymKeyAlias` — 从对称 KeyStore 获取的、在文件输入 MapReduce 作业前用于解密 HDFS 文件的对称密钥别名。
- i. `outputSymKeyAlias` — 从对称 KeyStore 获取的、在文件放入 HDFS 前用于加密 MapReduce 作业的输出文件的对称密钥别名。
- j. `inputFile` — 注明 HDFS 上加密文件的位置，此文件将通过 MapReduce 应用程序解密后作为作业的输入。



k. `outputFile` — 注明存放 MapReduce 应用程序输出结果的 HDFS 上的位置。

2. 参考以下情形：

```
KeyProviderConfig encryptionKeyProviderConfig =
KeyStoreKeyProvider.getKeyProviderConfig(asymKSFilePub,
keyStoreTypeJKS, null, asymKSPassFilePub, true);
```

`KeyProviderConfig` 是 `KeyProviderCryptoContextProvider` 的子类。`KeyProviderConfig` 类注明 `KeyProviderCryptoContextProvider` 类从 `TrustStore` 获取公共密钥需要的参数。在以上示例中，`getKeyProviderConfig` 方法获取以下信息：

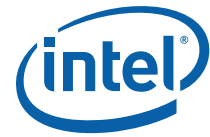
- `keyStoreFile` — 注明本地文件系统上 `TrustStore` 文件的位置。在以上示例中，这由 `asymKSFilePub` 变量定义。值必须遵守文件 URI 句法。比如，如果文件位于 `/tmp/test.keystore`，则值为 `file:///tmp/test.keystore`。
- `keyStoreType` — 注明使用的 keystore 类型。可能的值为 JCEKS 或 JKS。在以上示例中，因为执行的是对称加密，类型必须为 JCKS。
- `keyStorePassword` — 注明访问 `TrustStore` 的密码。由于已使用密码文件，这一字段的值为 `null`。
- `keyStorePasswordFile` — 注明本地文件系统上 `TrustStore` 密码文件的位置。在以上示例中，这由 `asymKSPassFilePub` 变量定义。值必须遵守文件 URI 句法。比如，如果文件位于 `/usr/lib/hadoop/keystore/test.truststore`，则值为 `file:///usr/lib/hadoop/keystore/test.truststore`。
- `sharedPassword` — `boolean` 值。如果密钥和 `TrustStore` 的密码相同，则值为 `true`。如果密钥和 `TrustStore` 的密码不同，则值为 `false`。在以上示例中，值为 `true`。

3. 参考以下情形：

```
KeyProviderConfig decryptionKeyProviderConfig =
KeyStoreKeyProvider.getKeyProviderConfig(clusterPrivateKeyStoreFile,
keyStoreTypeJKS, null, clusterPrivateKeyStorePasswordFile, true);
```

`KeyProviderConfig` 是 `KeyProviderCryptoContextProvider` 的子类。`KeyProviderConfig` 类注明 `KeyProviderCryptoContextProvider` 类从 `KeyStore` 获取私钥需要的参数。在以上示例中，`getKeyProviderConfig` 方法获取以下信息：

- `keyStoreFile` — 注明本地文件系统上对称 `KeyStore` 文件的位置。在以上示例中，这由 `clusterPrivateKeyStoreFile` 变量定义。值必须遵守文件 URI 句法。比如，如果文件位于 `/usr/lib/hadoop/keystore/test.keystore`，则值为 `file:///usr/lib/hadoop/keystore/test.keystore`。
- `keyStoreType` — 注明使用的 keystore 类型。可能的值为 JCEKS 或 JKS。在以上示例中，因为执行的是对称加密，类型必须为 JCKS。
- `keyStorePassword` — 注明访问对称 `KeyStore` 的密码。由于已使用密码文件，这一字段的值为 `null`。
- `keyStorePasswordFile` — 注明本地文件系统上非对称 `KeyStore` 密码文件的位置。在以上示例中，这由 `clusterPrivateKeyStorePasswordFile` 变量定义。比如，如果文件位于 `/usr/lib/hadoop/keystore/test.keystore.password`，则值为 `file:///usr/lib/hadoop/keystore/test.keystore.password`。
- `sharedPassword` — `boolean` 值。如果密钥和 keystore 的密码相同，则值为 `true`。如果密钥和 `TrustStore` 的密码不同，则值为 `false`。在以上示例中，值为 `true`。



## 4. 参考以下情形:

```
CredentialProtection credentialProtection = new CredentialProtection(
 encryptionKeyProviderConfig, encryptionKeyName,
 decryptionKeyProviderConfig, decryptionKeyName);
```

ClientProtection 类是 KeyProviderCryptoContextProvider 类的子类。ClientProtection 类是 KeyProviderConfig 对象的包装类，它包含执行作业配置文件非对称加密的必要信息。在以上示例中，ClientProtection 类通过使用以下参数调用构造函数来实例化。

- encryptionKeyProviderConfig — KeyProviderConfig 对象，包含了从 TrustStore 获得用于解密作业配置文件的公共密钥的信息。
- encryptionKeyName — TrustStore 中公共密钥的别名。
- decryptionKeyProviderConfig — KeyProviderConfig 对象，包含了从非对称 KeyStore 获得用于解密作业配置文件的私钥的信息。
- decryptionKeyName — 非对称 KeyStore 中 RSA 密钥的别名。

## 5. 参考以下情形:

```
FileMatches fileMatches = new
FileMatches(KeyContext.refer(inputSymKeyAlias,
Key.KeyType.SYMMETRIC_KEY, "AES", 128));
```

FileMatches 类使用常规表达以和 KeyContext 的文件名匹配。使用 KeyContext 的 refer 方法，可从 KeyStore 中获取对称密钥，该密钥将用来解密从 HDFS 获取的名称和 regex 匹配的文件。refer 方法将根据传递给方法的 ID 调用 KeyContext 的密钥。在以上示例中，以下参数会传递给方法:

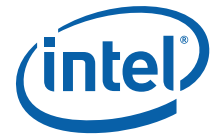
- keyalias — 从对称 KeyStore 获取的密钥别名，在 MapReduce 作业中作为输入数据被处理前用来解密 HDFS 数据。
- key.KeyType — 注明加密操作中使用的密钥类型。可能的值为 Key.KeyType.SYMMETRIC\_KEY, Key.KeyType.PUBLIC\_KEY, Key.KeyType.PRIVATE\_KEY, Key.KeyType.CERTIFICATE。在以上示例中，因为对文件执行的是对称加密，密钥类型必须为 SYMMETRIC\_KEY。
- cryptographicAlgorithm — 用于解密文件的对称算法。在以上示例中，加密算法为 AES。
- cryptographicLength — 用于解密文件的对称密钥大小。在以上示例中，对称密钥为 128 位。

## 6. 参考以下代码块:

```
KeyProviderConfig keyProviderConfig =
KeyStoreKeyProvider.getKeyProviderConfig(symKSFile, keyStoreTypeJCEKS,
null, symKSPassFile, false);
```

KeyProviderConfig 是 KeyProviderCryptoContextProvider 的子类。KeyProviderConfig 类注明 KeyProviderCryptoContextProvider 类从对称 KeyStore 获取私钥需要的参数。对称密钥用于解密从 HDFS 获取的数据。在以上示例中，getKeyProviderConfig 方法获取以下信息:

- a. keyStoreFile — 注明本地文件系统中对称 KeyStore 文件的位置。在以上示例中，这由 symKSFile 变量定义。值必须遵守文件 URI 句法。比如，如果文件位于 */usr/lib/hadoop/keystore/test.keystore*，则值为 *file:///usr/lib/hadoop/keystore/test.keystore*。
- b. keyStoreType — 注明使用的 keystore 类型。可能的值为 JCEKS 或 JKS。在以上示例中，因为执行的是对称加密，类型必须为 JCEKS。



- c. `keyStorePassword` — 注明访问对称 KeyStore 的密码。由于已使用密码文件，这一字段的值为 `null`。
- d. `keyStorePasswordFile` — 注明本地文件系统上对称 KeyStore 密码文件的位置。在以上示例中，这由 `symKSPassFile` 变量定义。比如，如果文件位于 `/usr/lib/hadoop/keystore/test.keystore.password`，则值为 `file:///usr/lib/hadoop/keystore/test.keystore.password`。
- e. `sharedPassword` — `boolean` 值。如果密钥和 keystore 的密码相同，则值为 `true`。如果密钥和 TrustStore 的密码不同，则值为 `false`。在以上示例中，值为 `false`。

7. 参考以下代码块：

```
KeyProviderCryptoContextProvider.setInputCryptoContextProvider(jobConf, fileMatches, true, keyProviderConfig, credentialProtection);
```

`setInputCryptoContextProvider` 方法指定 `KeyProviderCryptoContextProvider` 类用来解密从 HDFS 获得的数据所需要的选项、对象和数据。方法具有以下参数：

- `jobConf` — 用于存储解密和加密数据的关键信息的作业配置文件。
- `fileMatches` — 根据文件名是否和常规表达匹配，以确定用于解密从 HDFS 获得的数据的密钥。
- `clientSide` — `boolean` 值，用于注明在客户端提交作业到 JobTracker 前，对称密钥是否储存在作业配置文件中。如果是 `true`，则对称密钥储存在作业配置文件中。如果是 `false`，则对称密钥没有储存在作业配置文件中。
- `keyProviderConfig` — `KeyProviderConfig` 对象，包含了从对称 KeyStore 获得用于对称密钥的必要信息。
- `credentialProtection` — `CredentialProtection` 对象，包含了执行非对称加密来加密和解密作业配置文件的必要信息。

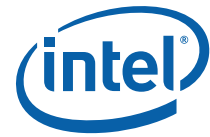
8. 参考以下代码块：

```
FileMatches fileMatchesForOutput = new
FileMatches(KeyContext.refer(outputSymKeyAlias, KeyType.SYMMETRIC_KEY,
Key.AES, 128));

KeyProviderCryptoContextProvider.setOutputCryptoContextProvider(jobConf, fileMatchesForOutput, true, keyProviderConfig, credentialProtection);
```

`FileMatches` 类使用常规表达以和 `KeyContext` 的文件名匹配。当 MapReduce 生成输出结果，且输出结果的文件名和常规表达匹配时，从 `KeyContext` 类中解析的密钥将在该文件被放入 HDFS 前被用来加密该文件。`setInputCryptoContextProvider` 方法指定 `KeyProviderCryptoContextProvider` 类用来加密要放入 HDFS 的数据所需要的选项、对象和数据。





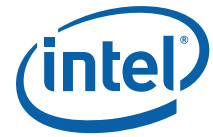
## Appendix A: Java 应用程序示例

---

### A.1 HDFS 应用程序执行加密

```
package com.intel.examples.crypto.client;
import java.io.InputStream;
import java.net.*;
import java.io.OutputStream;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.crypto.CryptoContext;
import org.apache.hadoop.io.crypto.Key;
import
org.apache.hadoop.mapreduce.cryptocontext.provider.KeyStoreKeyProvider;
import org.apache.hadoop.io.crypto.aes.AESCodec;
public class HDFSCryptoClientExample {
 public static void main(String[] args) throws Exception {
 /*
 * If no. of parameters is less than six, display the Usage command
and
 * exit.
 */
 if (args.length < 6) {
 System.err.println("Usage: \n"
 + "Encrypt: HDFSCryptoClientExample -e
keyStoreFile keyStorePasswordFile keyAlias inputFile outputFile \n"
 + "Decrypt: HDFSCryptoClientExample -d
keyStoreFile keyStorePasswordFile keyAlias inputFile outputFile \n");
 System.exit(2);
 }
 /*
 * Initialize variables from the vales passed in the command line
 */
 boolean encrypt = "-e".equals(args[0]);
 String keyStoreFile = args[1];
 String keyStorePasswordFile = args[2];
```





```

String keyAlias = args[3];
String inputFile = args[4];
String outputFile = args[5];

/*
 * Create a Configuration object
 */
Configuration conf = new Configuration();
/*
 * Instantiate and initialize KeyStoreConfig & KeyStoreKeyProvider
 * object.
 */
KeyStoreKeyProvider.KeyStoreConfig keyStoreParameters = new
KeyStoreKeyProvider.KeyStoreConfig(
 keyStoreFile, null, keyStorePasswordFile, "JCEKS", false);
KeyStoreKeyProvider keyProvider = new KeyStoreKeyProvider();
keyProvider.setConf(conf);
keyProvider.init(keyStoreParameters.toBytes());
/*
 * Fetch the key with the alias specified by the user.
 */
Key[] keys = keyProvider.getKeys(new String[] { keyAlias });
if (keys == null || keys.length <= 0) {
 System.out.println("No key entry for the alias " + keyAlias
 + " in " + keyStoreFile + " key store.");
 return;
}
/*
 * Instantiate the CryptoContext object and initialize it with the
key
 * fetched from the keystore.
 *
 *
 */
CryptoContext cryptoContext = new CryptoContext();
cryptoContext.setKey(keys[0]);
keys[0].setCryptographicLength(128);
/*
 * Instantiate the AES codec and set the CryptoContext object.
 */
AESCodec codec = new AESCodec();

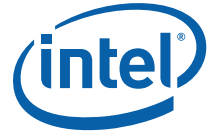
```



```

codec.setCryptoContext(cryptoContext);
((AESCCodec) codec).setConf(conf);
int totalRead = 0;
long start = System.currentTimeMillis();
FileSystem fsinput = FileSystem.get(conf);
FileSystem fsoutput = FileSystem.get(conf);
try {
 final Path in = new Path(inputFile);
 final Path out = new Path(outputFile);
 /*
 * If Input or output path is a local file get filesystem local
 * object
 */
 if (inputFile.startsWith("file:///")) {
 fsinput = FileSystem.getLocal(conf);
 }
 if (outputFile.startsWith("file:///")) {
 fsoutput = FileSystem.getLocal(conf);
 }
 byte[] buffer = new byte[1024 * 1024]; // 64k
 int read = 0;
 InputStream inStream = null;
 OutputStream outStream = null;
 try {
 /*
 * If -d switch is specified decrypt the input file
 */
 if (!encrypt) {
 inStream = codec.createInputStream(fsinput.open(in));
 outStream = fsoutput.create(out, true);
 } else { // if -e switch is specified encrypt the input file
 outStream =
 codec.createOutputStream(fsoutput.create(out,
 true));
 inStream = fsinput.open(in);
 }
 /*
 * Read from InputStream and write to the OutputStream
 */
 while (0 < (read = inStream.read(buffer, 0, 64 * 1024))) {
 totalRead += read;
 }
 }
}

```



```

 outputStream.write(buffer, 0, read);
 }
} finally {
 try {
 if (inStream != null)
 inStream.close();
 } catch (Exception ex) {
 System.out.println("Error closing Input Stream");
 ex.printStackTrace();
 }
 try {
 if (outStream != null)
 outStream.close();
 } catch (Exception ex) {
 System.out.println("Error closing Output Stream");
 ex.printStackTrace();
 }
}
} catch (Throwable e) {
 e.printStackTrace();
} finally {
 try {
 if (fsinput != null)
 fsinput.close();
 ;
 } catch (Exception ex) {
 System.out.println("Error closing Input FileSystem
Stream");
 ex.printStackTrace();
 }
 try {
 if (fsoutput != null)
 fsoutput.close();
 ;
 } catch (Exception ex) {
 System.out.println("Error closing Output FileSystem
Stream");
 ex.printStackTrace();
 }
}
System.out.println("Total bytes read: " + totalRead);
System.out.println("Total Time Taken: "

```



```

 + (System.currentTimeMillis() - start) / 1000 + " s");
 }
}
}

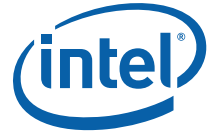
```

## A.2 MapReduce 应用程序执行加密

```

package com.intel.examples.crypto.mapreduce;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.compress.CompressionCodecFactory;
import org.apache.hadoop.io.crypto.Key;
import org.apache.hadoop.io.crypto.Key.KeyType;
import org.apache.hadoop.io.crypto.aes.AESCodec;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import
org.apache.hadoop.mapreduce.cryptocontext.provider.FileMatchCryptoContext
Provider;
import org.apache.hadoop.mapreduce.cryptocontext.provider.FileMatches;
import org.apache.hadoop.mapreduce.cryptocontext.provider.KeyContext;
import
org.apache.hadoop.mapreduce.cryptocontext.provider.KeyProviderCryptoConte
xtProvider;
import
org.apache.hadoop.mapreduce.cryptocontext.provider.KeyProviderCryptoConte
xtProvider.CredentialProtection;
import
org.apache.hadoop.mapreduce.cryptocontext.provider.KeyProviderCryptoConte
xtProvider.KeyProviderConfig;
import
org.apache.hadoop.mapreduce.cryptocontext.provider.KeyStoreKeyProvider;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
/*

```



```

 * This class expects the data on HDFS is encrypted.To load encrypted data
 into
 *
 *
 */
public class EncryptedWordCountExample {

 public static class TokenizerMapper extends
 Mapper<Object, Text, Text, IntWritable> {
 private final static IntWritable one = new IntWritable(1);
 private Text word = new Text();

 @Override
 public void map(Object key, Text value, Context context)
 throws IOException, InterruptedException {
 StringTokenizer itr = new StringTokenizer(value.toString());
 while (itr.hasMoreTokens()) {
 word.set(itr.nextToken());
 context.write(word, one);
 }
 }
 }

 public static class IntSumReducer extends
 Reducer<Text, IntWritable, Text, IntWritable> {
 private IntWritable result = new IntWritable();

 @Override
 public void reduce(Text key, Iterable<IntWritable> values,
 Context context) throws IOException, InterruptedException {
 int sum = 0;
 for (IntWritable val : values) {
 sum += val.get();
 }
 result.set(sum);
 context.write(key, result);
 }
 }

 public static void main(String[] args) throws Exception {

 Configuration conf = new Configuration();
 String[] otherArgs = new GenericOptionsParser(conf,
 args).getRemainingArgs();
 }
}

```



```

/*
 * If no. of parameters is less than six, display the Usage command and
 * exit.
 */
if (otherArgs.length < 10) {
 System.err.println("Usage: \n"
 + "EncryptedWordCountExample AsymKSFilePub AsymKSPassFilePub
 AsymKeyPvtFileName AsymPvtKeyAlias AsymPubKeyAlias SymKSFile
 SymKSPassFile inputSymKeyAlias outputSymKeyAlias inputFile outputFile
 \n");
 System.exit(2);
}

/*
 * Initialize variables from the vales passed in the command line
 */
System.out.println(args[0]);
System.out.println(args[1]);

String asymKSFilePub = args[0];
String asymKSPassFilePub = args[1];
String asymKeyPvtFileName = args[2];
String asymPvtKeyAlias = args[3];
String asymPubKeyAlias = args[4];
String symKSFile = args[5];
String symKSPassFile = args[6];
String inputSymKeyAlias = args[7];
String outputSymKeyAlias = args[8];
String inputFile = args[9];
String outputFile = args[10];

/*
 * Configuring the Job object
 */
Job job = new Job(conf, "Encrypted WordCount");
job.setJarByClass(EncryptedWordCountExample.class);
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(IntSumReducer.class);
job.setReducerClass(IntSumReducer.class);

```



```

job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path(inputFile));
FileOutputFormat.setOutputPath(job, new Path(outputFile));

/*
 * Configuring the JobConf object
 */
JobConf jobConf = (JobConf)job.getConfiguration();

/*
 * The JKS Keystore is used for asymmetric Key encryption.
 * The JCEKS Keystore is used for symmetric key encryption i.e.
encryption of data on HDFS
 *
 */
String keyStoreTypeJKS = "JKS";
String keyStoreTypeJCEKS = "JCEKS";

/*
 * Path for Private Key. The file name asymKeyPvtFileName will be
appended to this path.
 * The code assumes that the password file will have ".passwords" as
a suffix to the asymKeyPvtFileName name.
 */
String dirAtServer = "/usr/lib/hadoop/keystore/";
String clusterPrivateKeyStoreFile = "file:/// " + dirAtServer +
asymKeyPvtFileName;
String clusterPrivateKeyStorePasswordFile =
clusterPrivateKeyStoreFile+".passwords";

String encryptionKeyName = asymPubKeyAlias;
String decryptionKeyName = asymPvtKeyAlias;

// key provider for encryption the parameters and secrets
KeyProviderConfig encryptionKeyProviderConfig =
 KeyStoreKeyProvider.getKeyProviderConfig(
 asymKSFilePub, keyStoreTypeJKS, null, asymKSPassFilePub,
true);

```



```

// key provider for decrypt the parameters and secrets
KeyProviderConfig decryptionKeyProviderConfig =
 KeyStoreKeyProvider.getKeyProviderConfig(
 clusterPrivateKeyStoreFile, keyStoreTypeJKS, null,
 clusterPrivateKeyStorePasswordFile, true);

CredentialProtection credentialProtection = new CredentialProtection(
 encryptionKeyProviderConfig, encryptionKeyName,
 decryptionKeyProviderConfig, decryptionKeyName);

//set input codec class
List<Class> classes = new ArrayList<Class>();
classes.add(AESCodec.class);
 CompressionCodecFactory.setCodecClasses(jobConf, classes);

//set input key provider

// File matches for choose context for files
FileMatches fileMatches = new
FileMatches(KeyContext.refer(inputSymKeyAlias, Key.KeyType.SYMMETRIC_KEY,
"AES", 128));

KeyProviderConfig keyProviderConfig =
 KeyStoreKeyProvider.getKeyProviderConfig(
 symKSFile, keyStoreTypeJCEKS, null, symKSPassFile, false);

KeyProviderCryptoContextProvider.setInputCryptoContextProvider(jobConf,
 fileMatches, true, keyProviderConfig,
 credentialProtection);

//encrypt and decrypt intermediate map output by a key derived by a
password
 jobConf.setMapOutputCompressorClass(AESCodec.class);

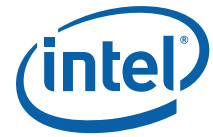
 FileMatches fileMatchesForMap = new
FileMatches(KeyContext.derive("12345678"));

FileMatchCryptoContextProvider.setMapOutputCryptoContextProvider(jobConf,
fileMatchesForMap, null);

/*

```





```
*
* Code for writing encrypted output the HDFs
*
*/

//set output codec class

org.apache.hadoop.mapred.FileOutputFormat.setOutputCompressorClass(jobConf, AESCodec.class);

//set output key provider
FileMatches fileMatchesForOutput = new
FileMatches(KeyContext.refer(outputSymKeyAlias, KeyType.SYMMETRIC_KEY,
Key.AES, 128));

KeyProviderCryptoContextProvider.setOutputCryptoContextProvider(jobConf,
fileMatchesForOutput, true, keyProviderConfig,
credentialProtection);

System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```