# Apache Ranger概述

**随着Hadoop生态系统越来越成熟，其现在已经能够支持完整的数据湖**（**Data Lake，是指来自多个不同来源的结构化或者非结构化数据，有选择的被收集在多个数据库或者存储系统上**）架构。企业在Hadoop系统上可以在多客户环境中运行多个工作负载。由于需要支持多用户访问企业数据湖，而数据对于企业来说是重要的资产，为了保护这些各种不同类型的用户数据，专门应对大数据的分布式用户安全架构应用而生。

**Apache Ranger是用于Hadoop的集中式安全管理解决方案**，使管理员能够为HDFS和其它Hadoop平台组件创建和实施安全策略，并且为Hadoop的各个零部件提供细粒度的安全权限机制。

## Ranger技术概况

Ranger的特点和作用

1. 细粒度授权
2. 集中化审计日志管理和策略管理
3. 易于操作控制权限
4. 统一的操作平台Ranger Web UI
5. 全面支持Hadoop的组件

## Ranger构成

### 内部依赖

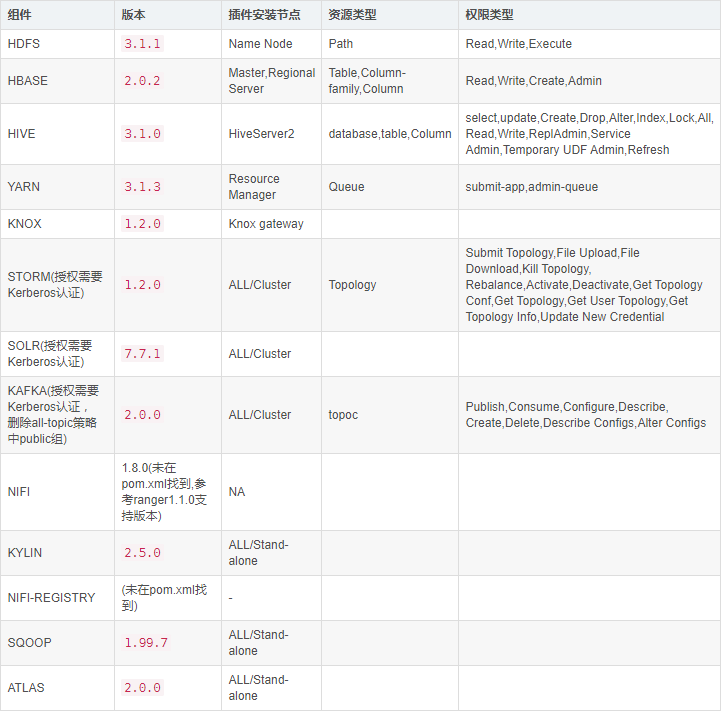


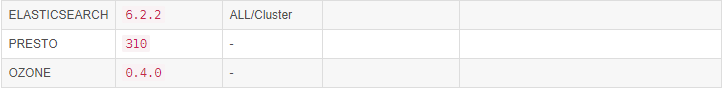
注：建议将审计存储在Solr和HDFS中。默认情况下，这两个选项都设置为ON。Solr提供了对最新日志进行索引和搜索的功能，而HDFS则用作更持久或更长期的存储。默认情况下，Solr用于索引前30天的审计日志。

### 内部组件



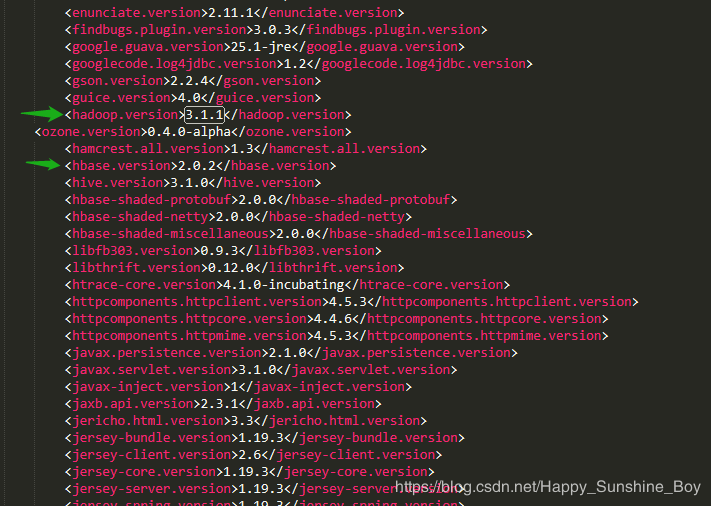
### Ranger插件





注： Apache atlas 是Hadoop社区为解决Hadoop生态系统的元数据治理问题而产生的开源项目，它为Hadoop集群提供了包括数据分类、集中策略引擎、数据血缘、安全和生命周期管理在内的元数据治理核心能力。

ranger-release-ranger-2.0.0/pom.xml 显示依赖组件版本



## Ranger架构

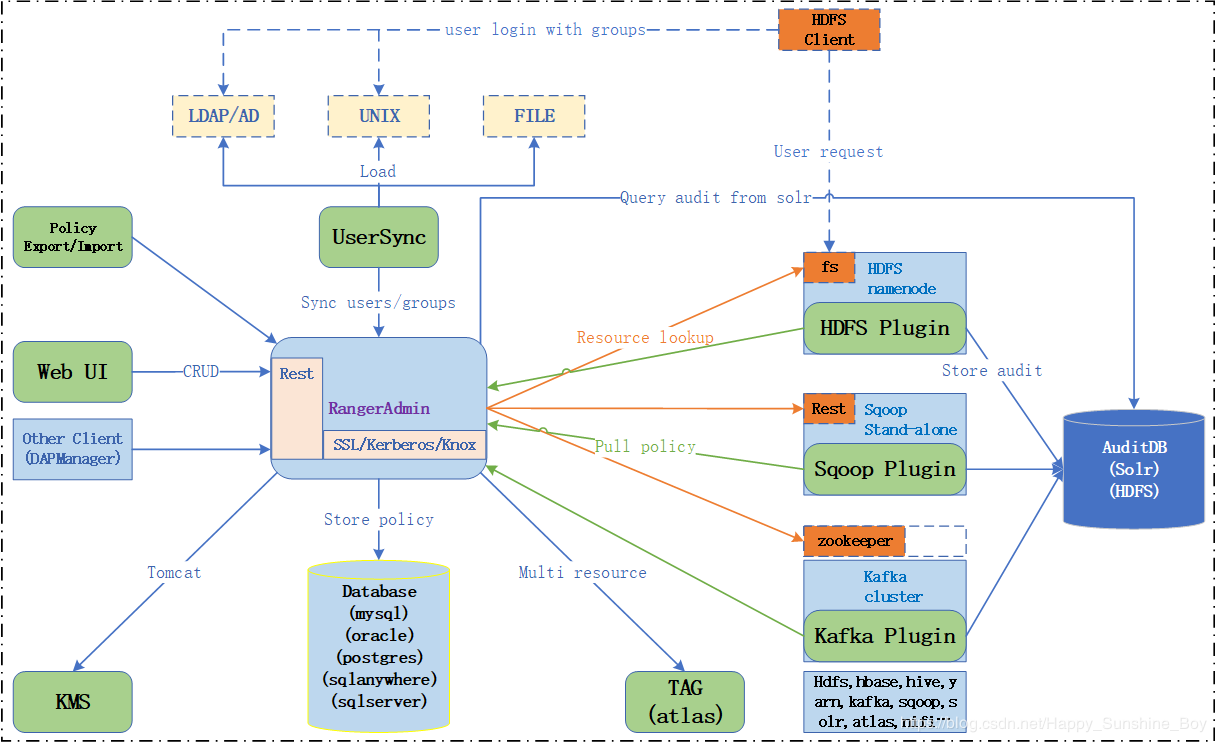
Ranger授权本质上是：通过读取安装组件时生成的配置文件及组件自带的jar包，通过hook方式调用各个组件服务达到权限管理。

在安装服务组件插件过程中，当执行./enable-xxx-plugin.sh时，主要执行了以下三个步骤：

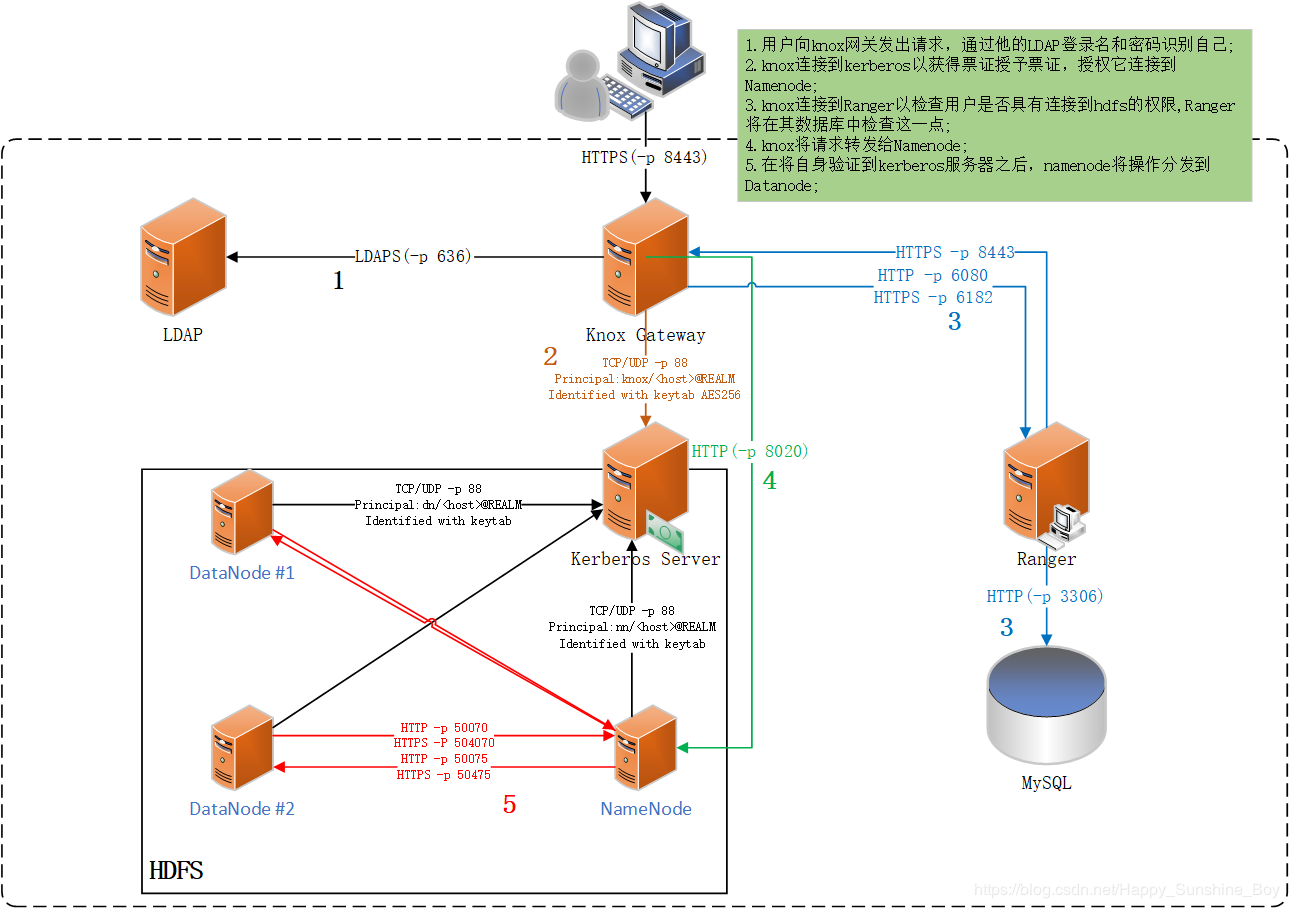
第一，将插件自带的conf更新到系统安装的服务conf下；

第二，将插件自带的lib更新到系统安装的服务lib下；

第三，将install.properties生成.xml文件，更新到系统安装的服务conf下；

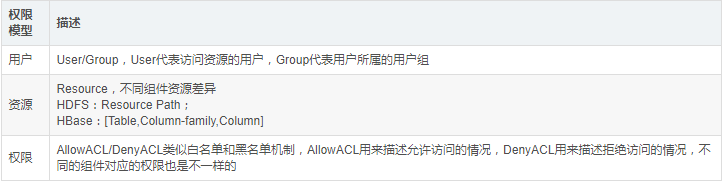


## Ranger + Knox + Kerberos 架构

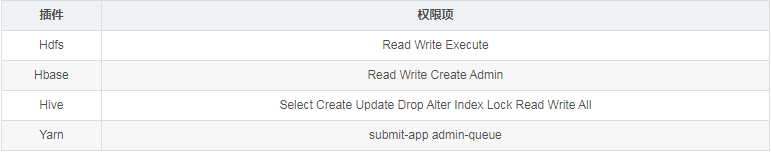


## 权限模型

访问权限无非是定义了“用户-资源-权限”这三者之间的关系，Ranger基于策略来抽象这种关系，进而延伸出自己的权限模型。



下表列出了几种常见系统的模型实体枚举值。



Apache Ranger为了实现企业级Hadoop生态系统的综合安全权限管理，提供了一个全面的Hadoop集群的安全策略，它的定义提供了一个集中的平台，管理和授权所有安全策略一致的Hadoop生态系统的组件，所以Ranger设计了一个分散的结构。

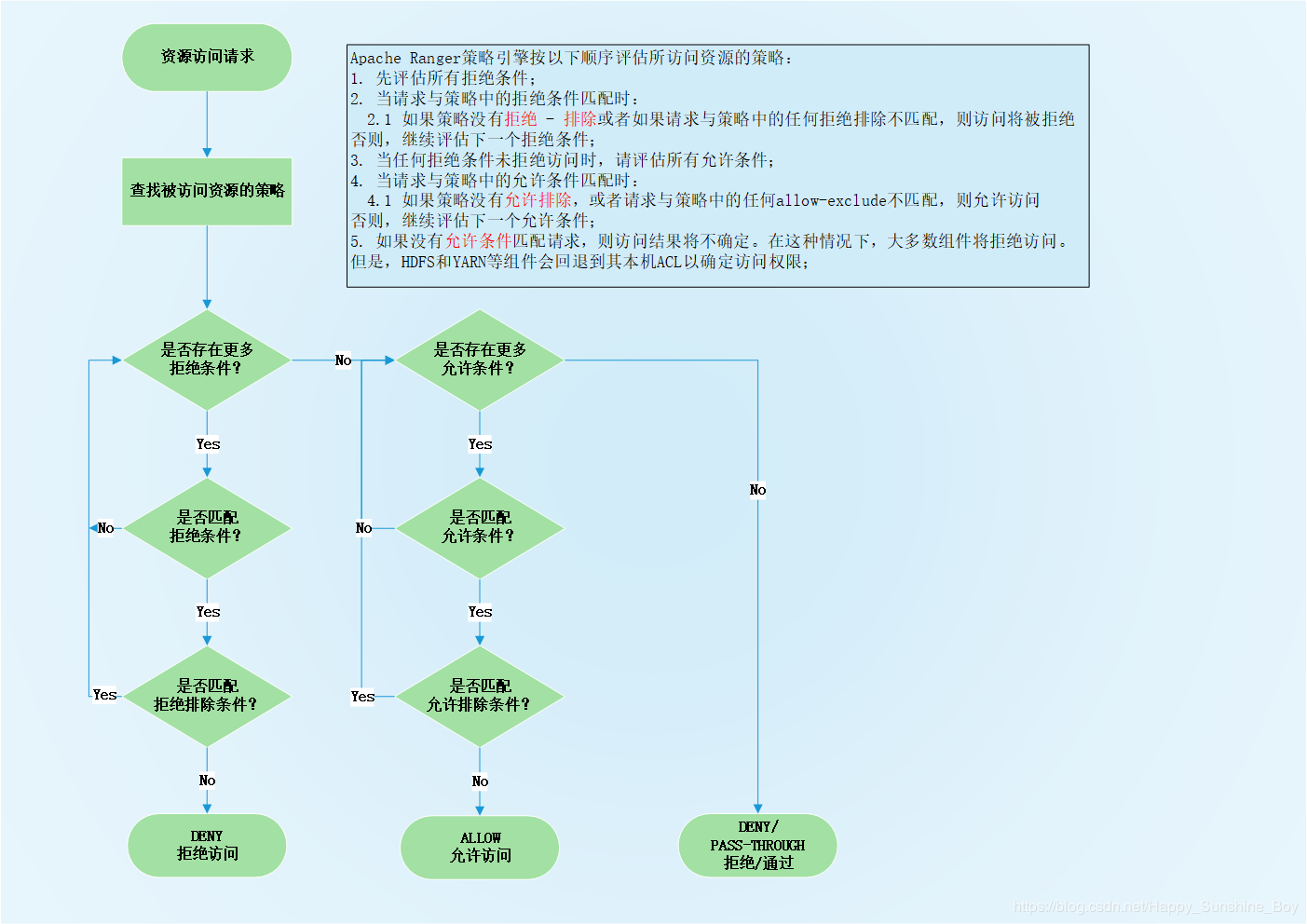
Ranger的分散结构使得它能在不需要使用某些组件的时候，可以不安装这个组件的插件，只要安装需要进行权限控制的组件即可。这种方式大大减轻了系统的复杂度，尤其是当需求只针对某一两个组件的时候，Ranger能很轻易的实现权限控制。

Ranger的安全认证机制有许多细节设计，包括Deny Condition以及Exclude策略，此策略启用之后，Ranger Admin服务将产生4种配置项：Allow Condition、Exclude From Allow Conditions、Deny Condition、Exclude From Deny Condition。这四种配置主要被分成两个部分，Allow与Deny,而Condition部分是整体控制，系统会识别其中配置的策略，对请求进行认证，但是如果需要更为细化的对配置进行精简，也可以在Exclude中配置，将把整体配置中的某些权限排除在外，实现更细粒度的权限控制。

例如，可以再Allow Condition中对用户组public设置所有权限，但是在这个用户组中，hdfs用户不应该拥有read权限，那么就在Exclude from Allow Conditions中将hdfs的read权限排除，使hdfs用户不能拥有public用户组所拥有的allow Condition配置。

Ranger还提供各类认证策略，认证策略包括Geo-based认证策略，基于Hive的Row-Level Filtering和Column-Masking验证，以及Tag同步认证等，这些策略能按照用户的需求对系统权限控制进行调配，具有非常好的适应性。

**Geo-based认证策略是基于IP地址的一个认证策略，可以针对集群中各个扮演不同角色的节点设置不同的权限，比如Master可以设置全部权限，而存储节点只能拥有数据仓库的读写权限。这个策略具有非常好的适应性，而且支持所有组件，是常用的认证策略之一。**而基于Hive的Row-Level Filtering和Column-Masking验证，能对Hive内的数据进行细粒度的过滤，也能对数据进行各类加密。



## Ranger应用场景

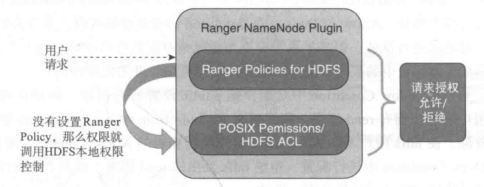
# Apache Ranger的安全认证配置

Ranger安装与部署

## 安全及访问权限控制机制

## Ranger集成HDFS的安全认证机制与配置

HDFS是所有Hadoop部署的核心部分，为了确保数据在Hadoop平台中得到保护，安全性需要嵌入HDFS层中，HDFS使用Kerberos身份验证以及使用POSIX标准权限HDFS ACL，或使用Apache Ranger（Ranger ACL）的授权进行保护。下图展示了Ranger策略如何工作，为HDFS提供联合授权模型，包括HDFS ACL和Ranger ACL（Access Control List 访问控制列表）。首先HDFS的Ranger插件检查Ranger策略，如果存在策略，则授予用户访问权限。如果Ranger中不存在策略，那么Ranger将默认使用HDFS中的本地权限模型。此联合模型适用于Ranger的HDFS和YARN服务。



对于其他服务（如Hive或HBase），Ranger作为唯一授权服务，这意味着只有Ranger策略生效，才能控制相应的权限。

hdfs-site.xml会修改如下配置：

<property>

<name>dfs.permissions.enabled</name>

<value>true</value>

</property>

<property>

<name>dfs.permissions</name>

<value>true</value>

</property>

<property>

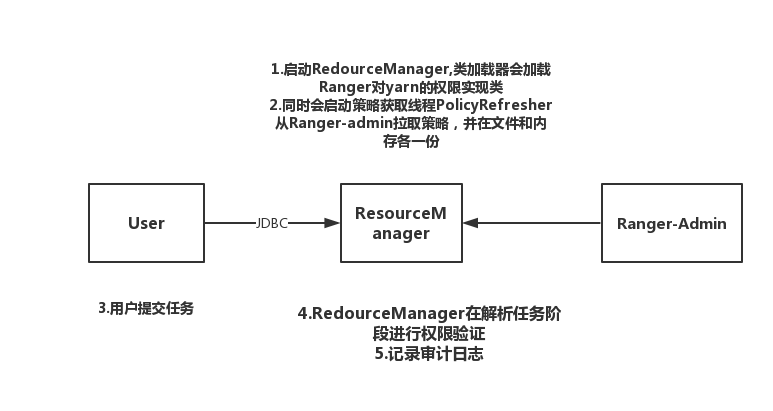
<name>dfs.namenode.inode.attributes.provider.class</name>

<value>org.apache.ranger.authorization.hadoop.RangerHdfsAuthorizer</value>

</property>

## Ranger集成YARN的安全认证机制与配置

加载过程：



yarn-site.xml会修改如下配置：

<property>

<name>yarn.acl.enable</name>

<value>true</value>

</property>

<property>

<name>yarn.authorization-provider</name>

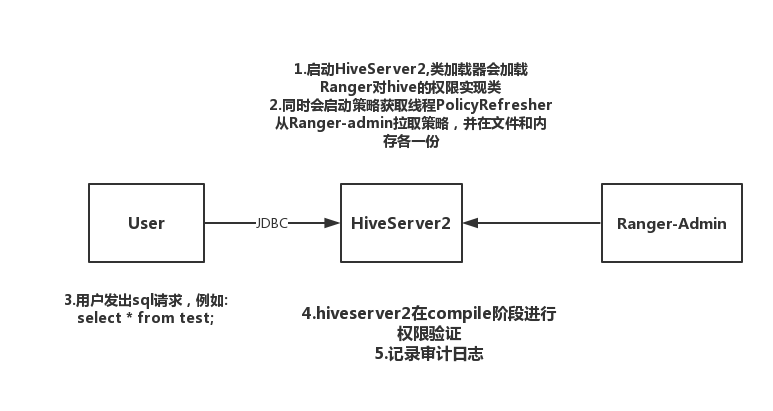
<value>org.apache.ranger.authorization.yarn.authorizer.RangerYarnAuthorizer</value>

</property>

## Ranger集成Hive的安全认证机制与配置

Ranger只支持Hive2.1.0及以上版本，HiveServer2（HS2）是一个服务器接口，使远程客户端能够执行Hive的查询并检索结果，HS2的实现基于Thrift RPC，是HiveServer的改进版本，支持多客户端并发和身份验证。

加载过程：



hiveserver2-site.xml会修改如下配置：

<property>

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

<value>true</value>

</property>

<property>

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

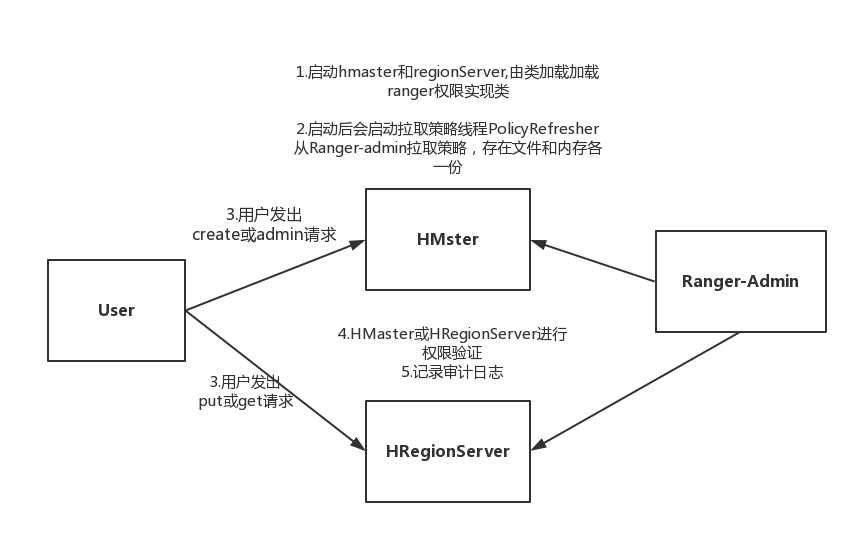
<value>org.apache.ranger.authorization.hive.authorizer.RangerHiveAuthorizerFactory</value>

</property>

## Ranger集成HBase的安全认证机制与配置

之前的Ranger组件只需要将插件安装到主节点上，而不需要安装到其它节点上，Ranger-Hbase插件必须作为coprocessors部署在Masger和Regionservers上，所以必须在所有节点上都安装Ranger-Hbase插件。

加载过程：



hbase-site.xml会修改如下配置：

<property>

<name>hbase.security.authorization</name>

<value>true</value>

</property>

<property>

<name>hbase.coprocessor.master.classes</name>

<value>org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor</value>

</property>

<property>

<name>hbase.coprocessor.region.classes</name>

<value>org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor</value>

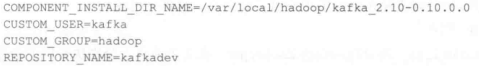
</property>

## Ranger集成Kafka的安全认证机制与配置

1. Ranger-Kafka解压与建立软连接
2. 修改配置Kafka插件的选项：

vim install.properties

修改以下配置项：



1. 修改kafak安装目录中的server.properties文件，添加如下属性：



1. 将Ranger-kafka插件所需的jar包复制到kafka的libs目录中，并且改变libs权限为kafka。



1. 在/etc/profile中添加kafka的配置文件的目录，并对环境变量进行刷新。



## Ranger集成Atlas的安全认证机制与配置

Atlas提供的数据治理能力和作为一个通过用的元数据存储器的牧师为了交换内外的Hadoop堆栈的元数据，Ranger提供了一个集中的用户界面，可以用来定义、执行和管理安全策略一致的Hadoop的堆栈中的所有组件，Atlas-Ranger在Ranger的安全执行策略中统一了Atlas的数据分类和元数据存储功能。

Atlas访问策略：

1. 基于分类的访问控制
2. 基于数据过期的访问策略
3. 特定位置的访问策略
4. 禁止数据集组合

## Ranger集成Storm的安全认证机制与配置

## Ranger集成Solr的安全认证机制与配置

1.初期准备

# Apache Ranger的功能配置

## Tag同步认证

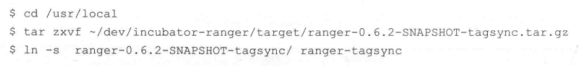
Ranger的Tag同步功能是Atlas与Ranger集成的一个非常重要的功能，Ranger如果能将Atlas中设置的Tag读取出来，并存储在Ranger服务器中，那么Ranger就可以对Tag设置的内容进行权限控制，这样就能实现更细粒度的权限控制，对系统有更完整、更方便的功能配置。

实现Ranger同步认证需要配置Ranger Tagsync服务，第一步是安装Tagsync插件。Tagsync插件能让Ranger读取Atlas中设置的Tag，甚至会自动为一些Tag设置权限。而对于另一些Tag，系统能读取Atlas设置的Tag含义，并保存在Ranger服务器中。

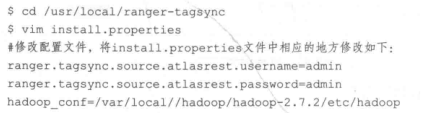
Tagsync插件安装步骤如下：

1. 配置Ranger Tagsync服务

首先应安装Tagsync插件，解压编译好的包，并设置链接。



修改安装配置：



上述配置中，username和password指的是Atals Web UI的登录账户名和密码，一般默认是admin/admin.

配置完成后，就可以安装插件并启动服务，服务启动之后，如果在Ranger的Web UI Setting中查询到新创建的用户ranger-tagsync，说明插件安装成功。



1. 添加Atlas Tag
2. 验证Tag同步

## 各类Policy验证

Ranger提供各种Policy验证，这些Policy验证大部分已经包含在系统中了，不需要安装额外的插件，这些验证机制能够细粒度的监控Ranger中的各种数据流动，用户提交的数据，并具有数据库的增删改查等功能，提供了各种系统权限监控的方式，以便于大数据的生态系统保持良好的用户权限分离。以下简单列举了几种Policy验证配置方式。

1. 配置Deny Condition以及Exclude选项
2. 配置Geo-based Policy验证

Geo-based Policy能对IP地址进行筛选，能对各个地址中的用户进行细分，实现细粒度的权限分离，能够控制各个不同的节点权限。

1. 配置基于Hive的Row-Level Filtering和Column-Masking验证

# Apache Ranger优化与性能分析

Ranger系统配置主要依赖于Web UI的各类配置，以及各个附加功能的使用，系统配置只要验证插件可以使用即可，但是对配置细节并没有做很好的优化，例如，HDFS插件配置中只是将HDFS中的某个特定目录下的某个权限赋予某个用户，使得这个用户不能使用未被赋予的权限。具体而言，如果只给用户赋予Write权限的话，他是无法使用Read权限的，那么那些没有被特定赋予权限的用户，则可以使用所有权限的。同样，某个用户如果只是被配置了特定的一个路径的权限，那么这个用户可以随意访问其他目录下的文件。

这种在配置没有完全包括所有情况的时候，系统对那些疏漏的用户是不存在权限控制的，同时，这种配置方式需要对每一个用户和每一个路径配置权限，这样做实在过于繁琐。

此外，比较常见的配置优化还有，在启动配置Deny Condition以及Exclude选项之后，系统对于权限的判断，如图所示，在这个流程中可以发现，系统先判断是否有Deny选项，再判断Allow，所以如果需要对系统进行更细致的权限控制的话，最好能对需要拒绝的用户采用Deny Condition。而对需要同意的用户使用Allow Condition。在配置不是特别复杂的情况下，能不使用Exclude选项就不使用，这在理论上会有一定的性能损耗。