如何使用Ranger增强权限管理

马晓琦

openLooKeng

https://openlookeng.io

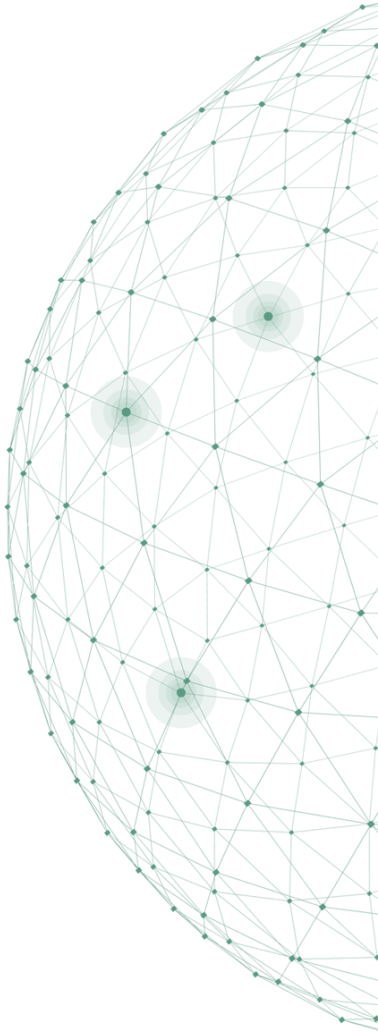
Content

Ranger简介

Ranger服务模型

Ranger插件介绍

Live Demo



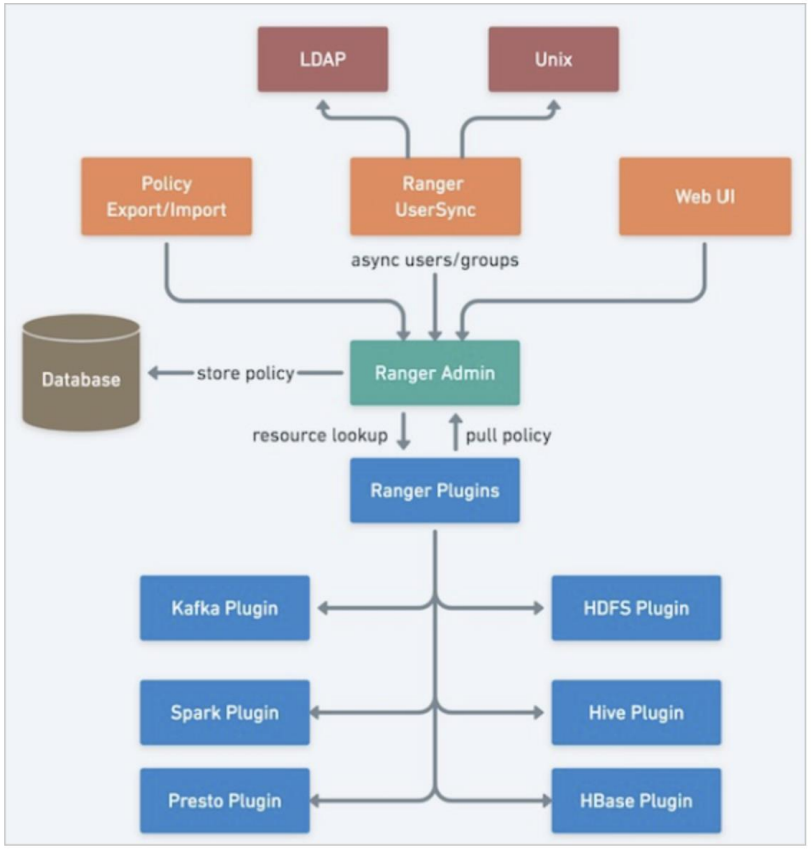
openLooKeng

https://openlookeng.io

Ranger简介

- Apache Ranger为Hadoop体系提供了统一的安全体系，包括细致的访问控制和统一的审计，能够监控和管理整个Hadoop平台的综合数据安全。
- Apache Ranger提供集中式的权限管理框架，可以对Hadoop生态中的HDFS/Hive/Yarn等组件进行细粒度的权限访问控制，并提供了Web UI方便管理员进行操作

- Ranger Admin
  用户可以创建和更新安全访问策略，这些策略被存储在数据库中。各个组件的Plugin定期对这些策略进行轮询
- Ranger Plugins
  Plugin嵌入在各个集群组件的进程里，是一个轻量级的Java程序
- Ranger UserSync
  Ranger提供了一个用户同步工具。您可以从Unix或者LDAP中拉取用户和用户组的信息



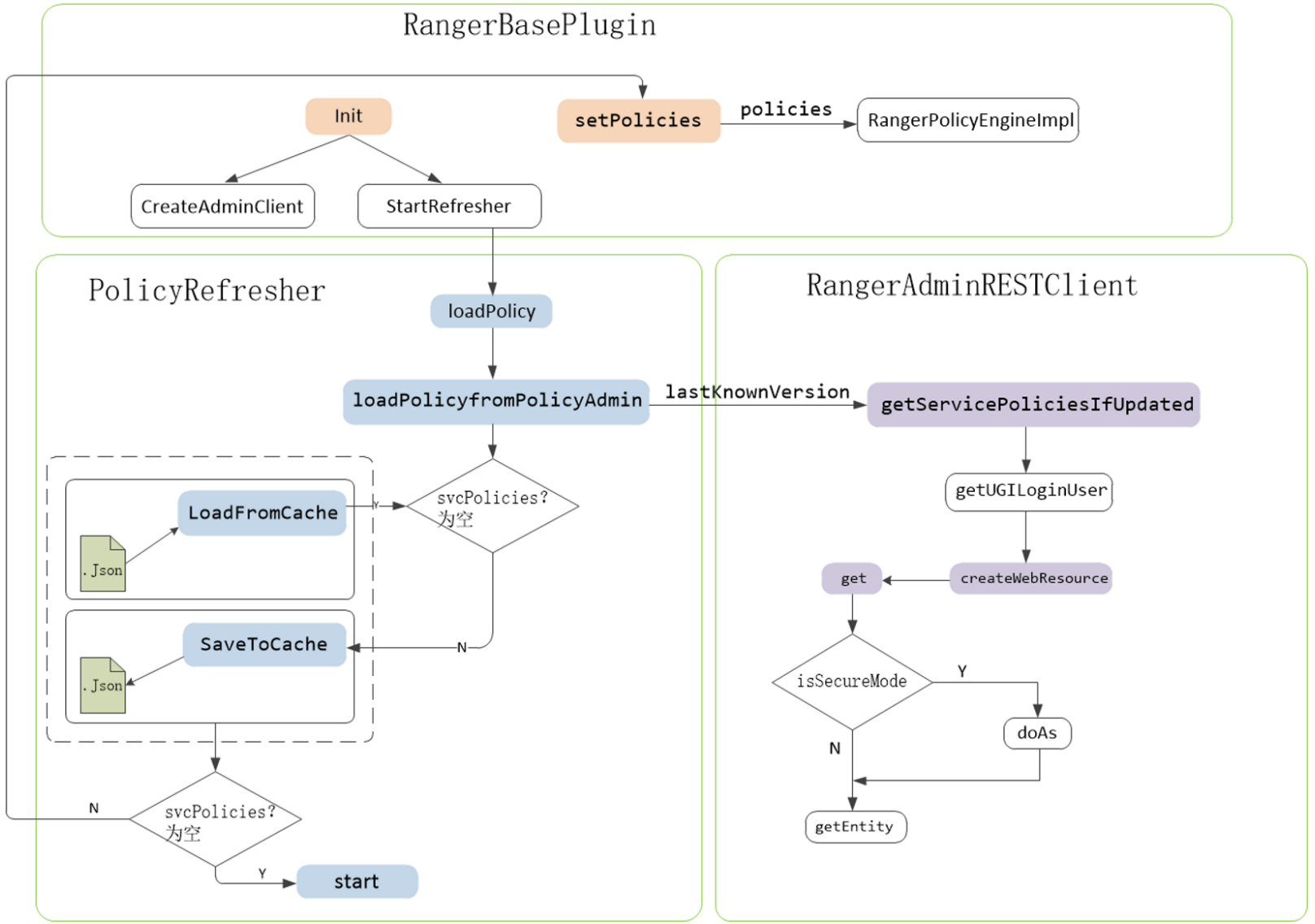
Ranger服务模型

- 权限模型：定义了”用户–资源–权限“这三者间的关系
  - **用户**：由User或Group来表达，User代表访问资源的用户，Group代表用户所属的用户组
  - **资源**：由Resource来表达，不同的组件对应的业务资源是不一样的，比如HDFS的File Path，HBase的Table
  - **权限**：由(AllowACL, DenyACL)来表达，类似白名单和黑名单机制，AllowACL用来描述允许访问的情况，DenyACL用来描述拒绝访问的情况。不同的组件对应的权限也是不一样的
- 服务类型
  - 创建一个JSON格式的文件，包含以下内容
    - 连接服务的配置(Service)：Username, Password
    - 资源(Resource)：Catalogs, Schemas, Tables, Columns
    - 访问类型(Access Type)：SELECT, INSERT, CREATE, DELETE
    - 其他自定义配置
  - 使用 Ranger Admin 提供的 REST API 向 Ranger 注册服务类型定义

```
curl -u admin:password -X POST -H "Accept: application/json" -H "Content
-Type: application/json" -d @service-defs/ranger-servicedef-openlookeng.
json http://ranger-admin-host:port/service/plugins/definitions
```

Ranger插件核心功能

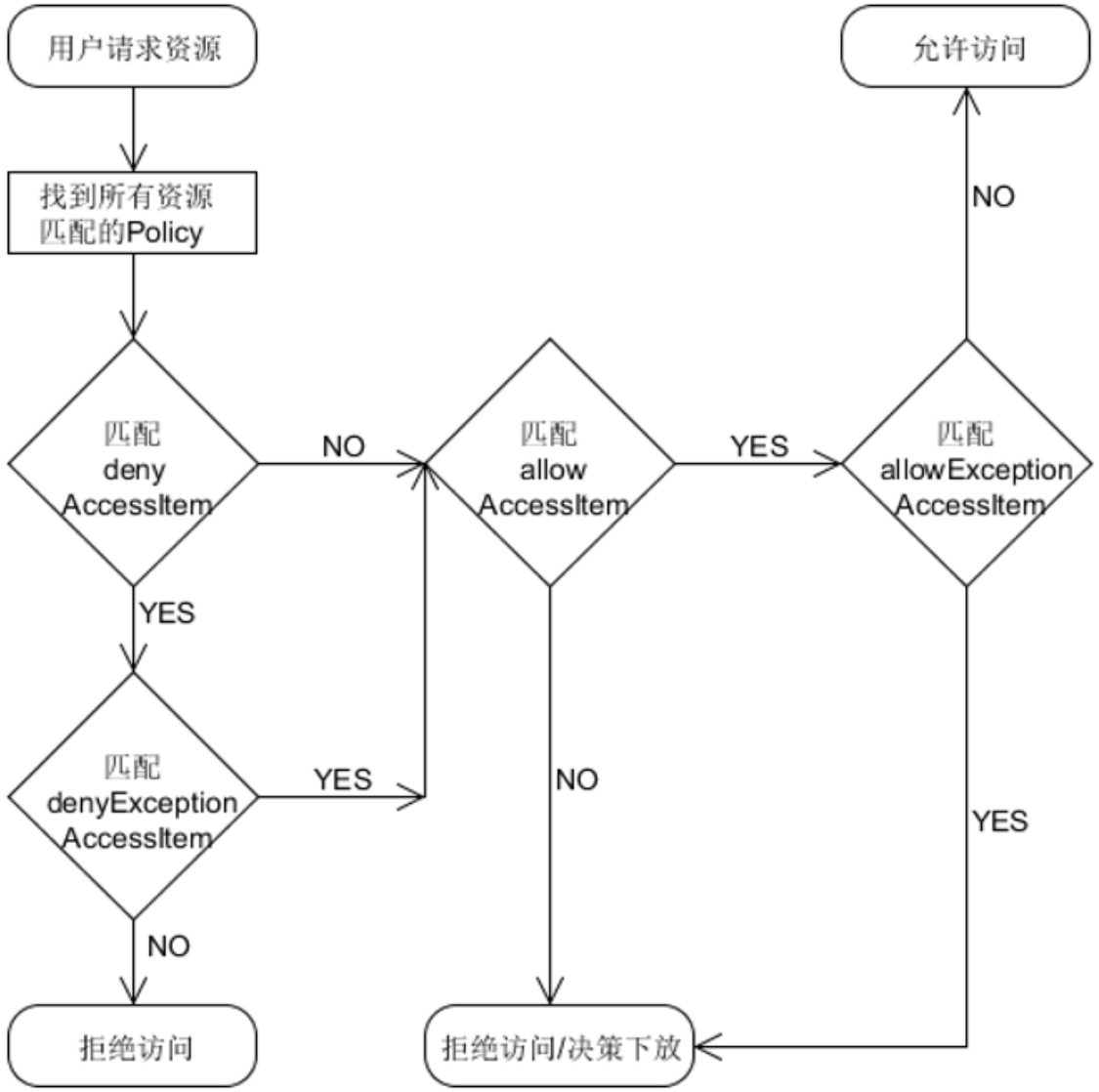
- 定期从Ranger Admin拉取策略
- 根据策略执行访问决策树
- 实时记录访问审计



➢ 插件主动到RangerAdmin拉取策略，插件将拉取到的策略保存到内存中的鉴权引擎，同时保存一份备份json文件在本地

Ranger插件核心功能

- 定期从Ranger Admin拉取策略
- 根据策略执行访问决策树
- 实时记录访问审计

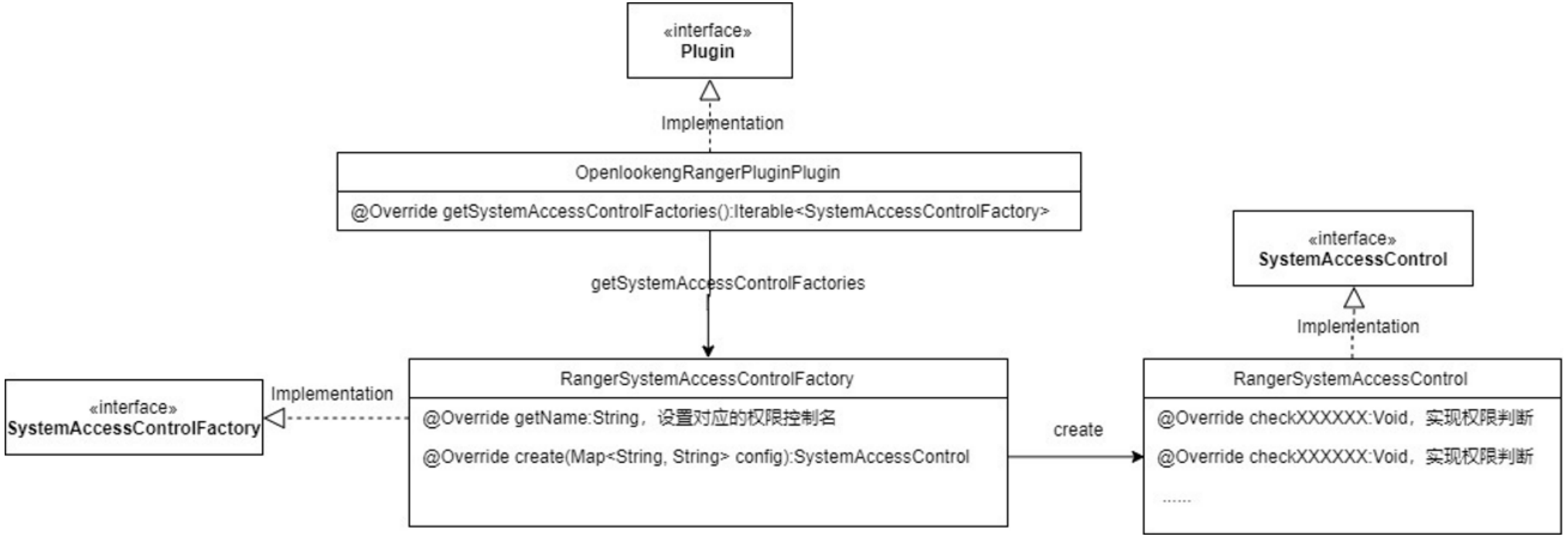


- 总体来说，这四组AccessItem的作用优先级由高到低依次是：denyException > deny > allowException > allow

- 总结一下就是黑名单优先级高于白名单，黑名单排除优先级高于黑名单，白名单排除优先级高于白名单。

Ranger插件开发

- 服务器端：驻留在Ranger Admin的代码/配置
- 应用程序端：驻留在应用程序服务中的代码，该代码调用Ranger服务并检查应用程序的最终用户是否有权访问他所请求的资源



openLooKeng

https://openlookeng.io

Ranger: 行过滤 & 列掩码

- 行过滤

```
lk> select * from mysql.db_test.table_test;
 id |        name         |  column_test   |       date_test
----+---------------------+----------------+-------------------------
  1 | Tom                 | l1AAGada110110 | 2011-11-11 11:11:33.000
  2 | Jack                | l19asf19119    | 2012-11-12 11:22:33.000
  3 | Rose                | l12Gfaf112112  | 2013-11-13 11:33:33.000
  4 | Adam                | l111kkk1HG11   | 2014-11-14 11:44:33.000
  5 | Haya                | 222afaf222222  | 2015-11-15 11:55:33.000
(5 rows)
```

Row Level Filter: **id >3**

```
lk> select * from mysql.db_test.table_test;
 id |        name         |  column_test   |       date_test
----+---------------------+----------------+-------------------------
  4 | Adam                | l111kkk1HG11   | 2014-11-14 11:44:33.000
  5 | Haya                | 222afaf222222  | 2015-11-15 11:55:33.000
(2 rows)

Query 20210127_030908_00015_dmcvy, FINISHED, 1 node
Splits: 17 total, 17 done (100.00%)
0:00 [2 rows, 0B] [9 rows/s, 0B/s]
```

- 列掩码

```
lk> select * from mysql.db_test.table_test;
 id |        name         |  column_test   |       date_test
----+---------------------+----------------+-------------------------
  1 | Tom                 | l1AAGada110110 | 2011-11-11 11:11:33.000
  2 | Jack                | l19asf19119    | 2012-11-12 11:22:33.000
  3 | Rose                | l12Gfaf112112  | 2013-11-13 11:33:33.000
  4 | Adam                | l111kkk1HG11   | 2014-11-14 11:44:33.000
  5 | Haya                | 222afaf222222  | 2015-11-15 11:55:33.000
(5 rows)
```

Column mask: **partial mask: show last 4**

```
lk> select * from mysql.db_test.table_test;
 id |        name         |  column_test   |       date_test
----+---------------------+----------------+-------------------------
  1 | Tom                 | XXXXXXXXX0110  | 2011-11-11 11:11:33.000
  2 | Jack                | XXXXXXX9119    | 2012-11-12 11:22:33.000
  3 | Rose                | XXXXXXXX2112   | 2013-11-13 11:33:33.000
  4 | Adam                | XXXXXXXHG11    | 2014-11-14 11:44:33.000
  5 | Haya                | XXXXXXXX2222   | 2015-11-15 11:55:33.000
(5 rows)

Query 20201115_134132_00997_9nn4b, FINISHED, 1 node
Splits: 17 total, 17 done (100.00%)
0:00 [5 rows, 0B] [66 rows/s, 0B/s]
```

openLooKeng

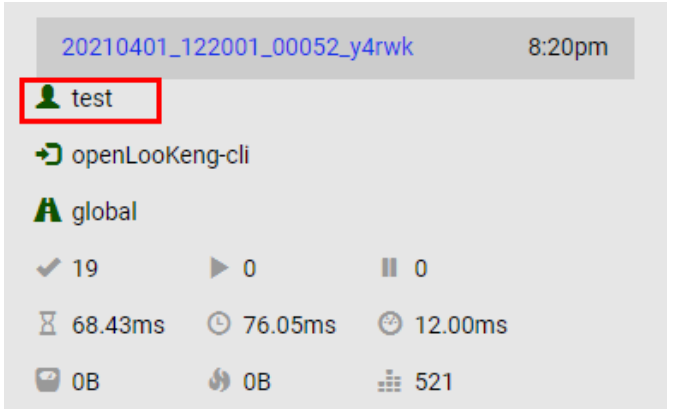
https://openlookeng.io

Ranger: 用户模拟

- 原始用户名: test@EXAMPLE.COM
- 映射后用户名: **test**

```
./hetu-cli \
  --server https://oracle19c:27778 \
  --krb5-config-path /etc/krb5.conf \
  --krb5-principal test@EXAMPLE.COM \
  --krb5-keytab-path /etc/openlookeng/test.keytab \
  --krb5-remote-service-name HTTP \
  --keystore-path /etc/keystore.jks \
  --keystore-password keystore_password
```

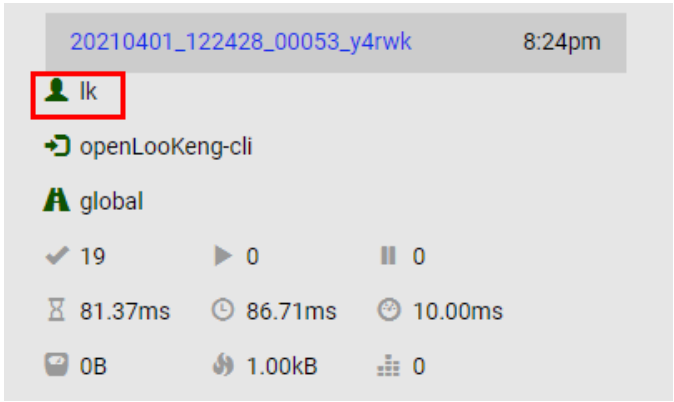
User name mapping



- 用户模拟: **lk** (用户“test” 能够模拟用户 “lk” ， 因此具有 “lk” 用户的所有权限)

```
./hetu-cli \
  --server https://oracle19c:27778 \
  --krb5-config-path /etc/krb5.conf \
  --krb5-principal test@EXAMPLE.COM \
  --krb5-keytab-path /etc/openlookeng/test.keytab \
  --krb5-remote-service-name HTTP \
  --keystore-path /etc/keystore.jks \
  --keystore-password keystore_password --user lk
```

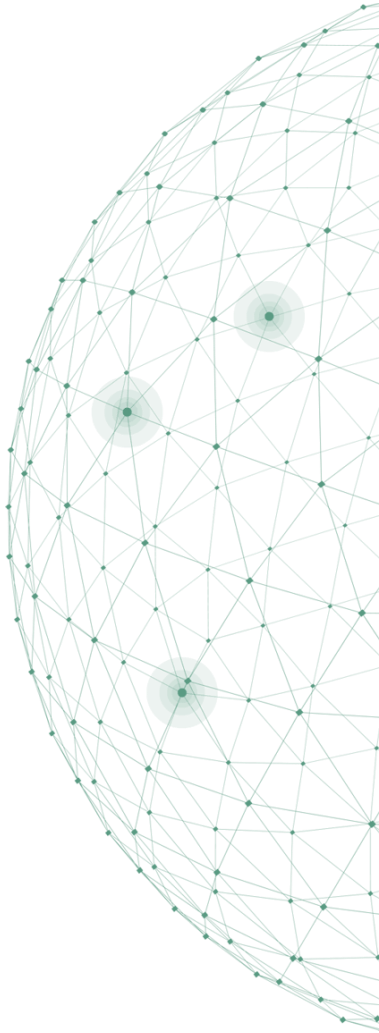
User impersonation



openLooKeng

https://openlookeng.io

Live Demo



openLooKeng

https://openlookeng.io

openLooKeng, Big Data Simplified

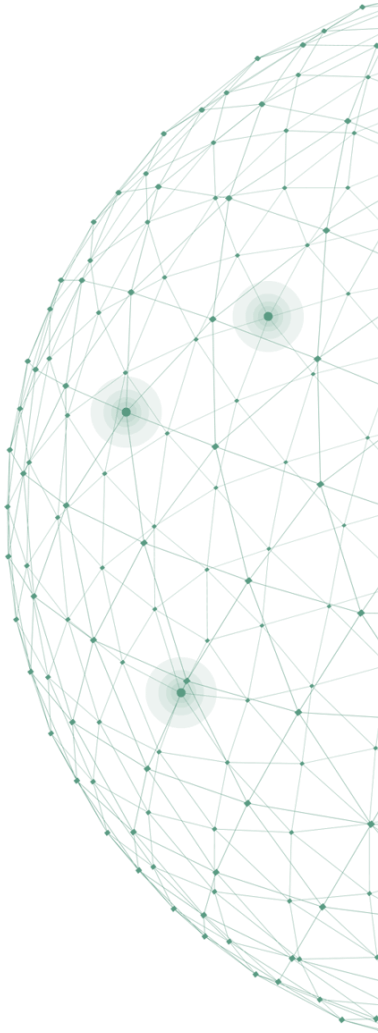
openLooKeng微信公众号

openLooKeng微信小助手

openLooKeng

https://openlookeng.io

Thank you!



openLooKeng

https://openlookeng.io