CK 权限管理方案

1 原则

1. 不同集群的管理方式保持一致（包括测试集群）；
2. 不同集群同一个业务的账号密码保持一致，降低业务切换的风险；
3. 充分利用rbac的机制，提升授权管理效率，降低管理复杂度；
4. 默认不含有字典（Dictionary）的操作权限，如需要，需单独赋权；

2 细则

2.1 acceess management

我们用default账号管理集群，开启access management后拥有所有权限；

1. config.xml开启access_control_path；
2. default user开启access_management并设置密码；
3. cluster定义里集群间通信默认用的是default，需要同步配置密码。

2.2 monitor提供给监控系统使用

```
CREATE USER IF NOT EXISTS monitor IDENTIFIED WITH double_sha1_hash BY
'90A785DE5EE0E3A96484DCF4CB94964CB7DBCC32' on CLUSTER X;
--- 默认用户都有select * from system.clusters权限，所以不用赋权
```

2.3 ROLE USER

由于role绑定user时不能指定database，所以多个database的相同权限不能定义到一个role上，所以这部分不用role，直接赋权给user，global的权限用role来授权；

```
--- 创建可复用的role biz_admin
CREATE ROLE IF NOT EXISTS jasong_admin ON CLUSTER X;
```

```
--- 授予global级别的权限给biz_admin
GRANT ON CLUSTER X CREATE TEMPORARY TABLE, SOURCES ON *.* TO jasong_admin;



--- 创建user
--- 生成密码（已存在的业务user用之前的密码）： head -c100 /dev/random | md5sum | head -c16
CREATE USER IF NOT EXISTS test ON CLUSTER X IDENTIFIED WITH double_sha1_password BY '1234';



--- 给user授予某个database下的权限
--- GRANT ON CLUSTER X SHOW, SELECT, INSERT, ALTER, CREATE DATABASE, CREATE TABLE, CREATE VIEW, DROP, TRUNCATE, OPTIMIZE, SYSTEM MERGES, SYSTEM TTL MERGES, SYSTEM FETCHES, SYSTEM MOVES, SYSTEM SENDS, SYSTEM REPLICATION QUEUES, SYSTEM SYNC REPLICA, SYSTEM RESTART REPLICA, SYSTEM FLUSH DISTRIBUTED ON db.* TO test;
--- 相当于 GRANT ALL ON db.* TO test;
GRANT ON CLUSTER X
 SHOW,
 SELECT,
 INSERT,
 ALTER,
 CREATE DATABASE, CREATE TABLE, CREATE VIEW,
 DROP,
 TRUNCATE,
 OPTIMIZE,
 SYSTEM MERGES, SYSTEM TTL MERGES, SYSTEM FETCHES, SYSTEM MOVES, SYSTEM SENDS, SYSTEM REPLICATION QUEUES, SYSTEM SYNC REPLICA, SYSTEM RESTART REPLICA, SYSTEM FLUSH DISTRIBUTED
ON db.* TO test;


--- 把jasong_admin的权限赋给user
GRANT ON CLUSTER X jasong_admin TO test;
```

3 CK创建新用户脚本

create_user.sh

```
#!/bin/bash
```

```
set -e

if [ $# -lt 3 ]; then
    echo "usage: $0 host http_port admin_password user [comma_sep_dbs]"
    exit 1
fi

#type mysql

host=$1
port=$2
admin_password=$3
user=$4
if [ $# -ge 5 ]; then
        dbstr=$5
else
        dbstr=$user
fi
dbs=$(echo $dbstr | tr "," "\n")

admin_user=default
user_admin_role=biz_admin

get_cluster_query="SELECT DISTINCT cluster FROM system.clusters WHERE cluster !=
'system_cluster'"
# 1. get cluster name
admin_server_url="http://${admin_user}:${admin_password}@${host}:${port}/"
cluster=$(curl -sSL "${admin_server_url}" -d "${get_cluster_query}")

# 2. create biz_admin role for one
biz_admin_role_exist_query="select count(*) from system.roles where name =
'${user_admin_role}';"
biz_admin_exist=$(curl -sSL "${admin_server_url}" -d
"${biz_admin_role_exist_query}")
if [ "X${biz_admin_exist}" = "X" ] || [ ${biz_admin_exist} -eq 0 ]; then
    create_biz_admin_query="CREATE ROLE IF NOT EXISTS \"${user_admin_role}\" ON
CLUSTER \"${cluster}\""
    grant_for_biz_admin_query="GRANT ON CLUSTER \"${cluster}\" CREATE TEMPORARY
TABLE, SOURCES ON *.* TO \"${user_admin_role}\""

    curl -sSL "${admin_server_url}" -d "${create_biz_admin_query}"
    curl -sSL "${admin_server_url}" -d "${grant_for_biz_admin_query}"
else
    echo "role ${user_admin_role} already exist!"
fi

# 3. generate password for new user
```

```
user_password=$(head -c100 /dev/random | md5sum | head -c15 | base64)
# use to set a pre-generated password
#user_password="1234"
passwd_hash=$(echo -n "${user_password}" | openssl sha1 -binary | openssl sha1 -
hex | awk '{print "*"toupper($0)}' | awk '{print $2}')
echo "user: ${user} password: ${user_password} hash:${passwd_hash}"


create_user_query="CREATE USER IF NOT EXISTS \"${user}\" ON CLUSTER
\"${cluster}\" IDENTIFIED WITH double_sha1_hash BY '${passwd_hash}';"
grant_global_query="GRANT ON CLUSTER \"${cluster}\" ${user_admin_role} TO
\"${user}\""

# 4. create new user, new database, and grant privileges to it
curl -sSL "${admin_server_url}" -d "${create_user_query}"
curl -sSL "${admin_server_url}" -d "${grant_global_query}"
for db in ${dbs[@]}; do
        grant_db_query="GRANT ON CLUSTER \"${cluster}\" SHOW, SELECT, INSERT,
ALTER, CREATE DATABASE, CREATE TABLE, CREATE VIEW, DROP, TRUNCATE, OPTIMIZE,
SYSTEM MERGES, SYSTEM TTL MERGES, SYSTEM FETCHES, SYSTEM MOVES, SYSTEM SENDS,
SYSTEM REPLICATION QUEUES, SYSTEM SYNC REPLICA, SYSTEM RESTART REPLICA, SYSTEM
FLUSH DISTRIBUTED ON \"${db}\".* TO \"${user}\";"
        create_db_query="CREATE DATABASE IF NOT EXISTS \"${db}\" ON CLUSTER
\"${cluster}\""
        curl -sSL "${admin_server_url}" -d "${grant_db_query}"
        curl -sSL "${admin_server_url}" -d "${create_db_query}"
done

#clickhouse client -m -h $host --port $port -u ${admin_user} --
password=${admin_password} --query "${create_user_query}"
curl -sSL "http://${user}:${user_password}@${host}:${port}/" -d "show databases"
&& echo "test new account success."

echo "created successfully!"
echo "cluster: ${cluster} user: ${user} password: ${user_password}"
```

4 字典（Dictionary）的赋权

```
-- Dictionary的访问权限，（DROP由于历史原因之前已经一并赋权）
GRANT ON CLUSTER KC0_CK_TS_01 CREATE DICTIONARY, dictGet ON mydb.* TO myuser;
-- 重新加载Dictionary的权限（一般不用）
GRANT ON CLUSTER KC0_CK_TS_01 SYSTEM RELOAD DICTIONARY ON *.* TO myuser;
```

5 只读账号的赋权

```
-- 参加好账号后给账号赋某个库的只读权限
GRANT ON CLUSTER \"${cluster}\" SHOW, SELECT ON "${db}".* TO "${user}";
-- 验证权限是否OK
SHOW GRANTS FOR "${user}";
```