USER LIST COLLECTOR

DESIGN DOCUMENT

REQUIREMENT :

To collect the mysql username and hostname from mysql database.

To get the specific grant privileges , which is the access grants for those users with their corresponding host in the Mysql database .

To Fetch the new added user and also the old dropped user from Mysql database .

To monitor the Mysql database userhost and their respective privileges from the Mysql database in a scheduled manner by resolving the above three steps and share the result as a CSV output file format as well as in a table format notified in mail.

DEPENDENCIES :

The user list is collected for multiple servers so it must be deployed in a centralised monitoring server . so from the monitoring node , the end db nodes will be connected to fetch the user info . we need password-less authentication from the monitor node to all the db servers which needs monitoring of the user list .
~/.my.cnf file should be enabled in the end db nodes .
Sendmail should be configured in the monitor node for getting mail alerts.

DEPENDENCY FILE :

DB SERVER FILE : mydbfile.txt

We need a text file as input which will act as a config file which consists of db related details for the job where the servername , ssh username , ssh hostip are given as input .

The names are given as follows separated by a space ,

Server_name1 ssh_user ssh_hosip

Sample_server1 root 10.0.2.13
Sample_server2 root 10.0.2.13

.

.
.
Likewise you may add . n no of servers but those servers should accept this dependency.

DEPENDENCY DIRECTORY:
DIRECTORY: mydir

A directory is created for maintaining the self generated files for validation .

The userlist info and the grants info are collected in files and stored inside the directory .
The files which are collected are self generated files and will be purged every week .
Set of four files are collected for each server and the files are old userhost and current userhost file , old_md5_grants and current_md5_grants .

Old files are maintained for the validation of newly added and dropped users from the mysql user table .
Old files are purged every week and current files become old files once the script is started and new current files will be generated for the db servers .

The files will be like the following example ,

UserHost_sample_server1---> contains current user host list
old_UserHost_sample_server1---> contains old user host list
Md5_sample_server1 ------> contains md5 of current user host ist .
Old_md5_sample_server1 -----> contains old md5 of user host list .

The above are the files for validation of the user list .
Then with the comparison of those files will be identified by diff command and it will be stored in the a file .

Changes -------> this file will contain the information of the newly added and dropped user in the server . this file is rotated according to the no of servers in the DBSERVER file . so this file will contain the last server's information.

CONFIG VARIABLES :

mail=("testing@gmail.com " “sample@gmail.com)----------→ EMAIL ID for the notification of mail.

server_path=/path/to/dependency/DBSERVER ----------→ #path of the servers file (config files) .
info_path=/path/to/dependency/directory----------→ #path of the USERINFO dir (config dir → used for internal computing)

The above three variables should be edited in the code for a proper job .

WORKFLOW :

here , i have given the overview of the workflow in this job ,

1) Every line in the DBSERVER file is iterated in a loop and the names user ip are processed in a different corresponding variable name .
2) Check for the user host file , if it exists then , move the file to old_UserHostfie.
3) It will open an ssh connection and will get the username and host name from mysql. user table in the remote server and will store it in the local as UserHost file .
4) The UserHost file which contains the user related information is iterated again in a loop and every user and host is taken out separately .
5) Check for md5 file , if the file present in the desired path then , mv the file to old_md5 file
6) Now again , the user host is converted into md5 value in a file (md5) file .
7) Now both the md5 files are validated in if condition whether it has the same data in it .
8) If yes , then no change function will be called .
9) If no , using diff command then differences between the old UserHost file and new UserHost file are taken out in the changes file . (note , that this process is taken on a single server .)
10) This below algorithm will be activated only when there is a change in the userhost file (i.e any new user are added or dropped in the dbserver)
11) Now the changes file is again iterated in a loop and each word is grepped and validated , and the added users and dropped users are calculated .
12) Here it goes with a short algorithm ,
    The output in the changes file is differentiated with the direction symbol (><) .
    the loop reads the line in the file as string by string .
    if the string is ">" then , the next two strings i+1 and i+2 are added to the user and added to the host .
    grant for the added user is fetched from the remote db and stored in a variable.
    add function called
    elif the string is "<" then , the next two strings i+1 and i+2 are dropped user name and dropped host name .
    drop function called .
    finally once the iteration of the changes file is completed then , no change function is called again .
    here add and drop functions are called for every user .

this way the algorithm works .

Nochange function is called , even if the userlist has changed or nochange in it .
functions :

note : files used in all mail functions are single file . here each file for each dbserver.

1) nochnage function : having html code for mailing file which is outputted to /tmp/output_table.csv , for building a table structure , so the table structure is built in this file . no change function will read the UserHost file completely in a loop and will get the grants for each user in a variable via a ssh connection . it will be outputted in a html table format as a html file and also in a csv format in a csv file . all the users are loaded in a single html table .

2) add function : add function is called whenever a user is added in the server . the grants of the user is also passed as argument along with the user and host name .

it will build a seperate table id outputted to the /tmp/output_table.csv for every single user , if 100 users are added , then 100 tables are generated. same as it is also loaded into a csv file as well .

3) drop function : drop function is called whenever a user is dropped from the server . plays as same as the add function but grants will not be provided since the user is dropped from the db server . Apart from that , this plays as equal as an add function .

mail :

two types of mail will be triggered ,

1) For sending the table structure , i.e the html file is passed in the web browser to show the file in the table format in the end output , so we will receive a mail in table structure and the structure holds the user , host grants .

2) For sending the csv file , a mail will be called which holds the csv in it .

Once the mail is sent , the next line in the dbserver file is read and processed for the next server .

this is the complete workflow of the script as an overview , but also it have many interesting algorithms ,

Please refer to the script for the reference .

FEATURES :

1) A complete monitoring of user management is done for production db servers via a monitor node .

2) Each user added or dropped in the db will be notified for every week .

3) Also a complete list of users and hosts in the db with their respective grant privileges are monitored and reported as a table structure as well as csv file .

4) includes total no of users in the server with server ip and server name is mentioned .

5) Each server details is sent separately to the users .

are some of the features .

Please check the design of the script