

BOOTCAMP ESPECIALIDAD GNU/LINUX (2023)

Lab 07 - Configuración de copias de seguridad en bases de datos

Pedro Antonio Mayorgas Parejo

20 de agosto de 2023

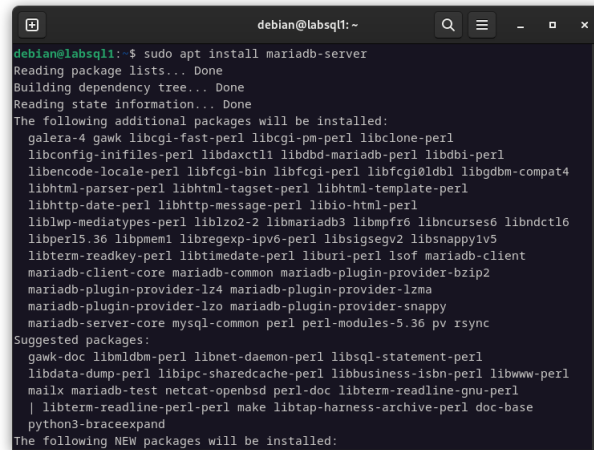
Índice

1	Instalación de Sistema Gestor de Base de Datos	3
2	Datos de la base de datos	5
3	Script de Bash para CRON	7

1. Instalación de Sistema Gestor de Base de Datos

Para la instalación del SGBD, debemos en ejecutar el siguiente comando en una distribución Debian GNU/Linux.

```
1 sudo apt install mariadb-server
```

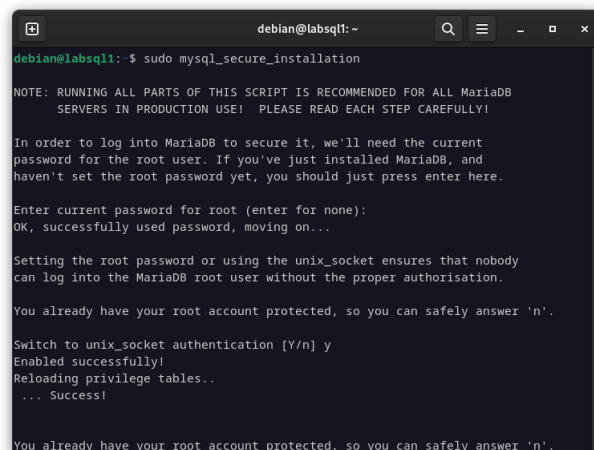


```
debian@labsql1: ~  
$ sudo apt install mariadb-server  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  galera-4 gawk libcgi-fast-perl libcgi-pm-perl libclone-perl  
  libconfig-inifiles-perl libdaxctl1 libdbd-mariadb-perl libdbi-perl  
  libencode-locale-perl libfcgi-bin libfcgi-perl libfcgi0ldbl libgdbm-compat4  
  libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl  
  libhttp-date-perl libhttp-message-perl libio-html-perl  
  liblwp-mediatypes-perl liblzo2-2 libmariadb3 libmpfr6 libncurses6 libndctl6  
  libperl5.36 libpmem1 libregexp-ipv6-perl libsigsegv2 libsnappy1v5  
  libterm-readkey-perl libtimedate-perl liburi-perl lsof mariadb-client  
  mariadb-client-core mariadb-common mariadb-plugin-provider-bzip2  
  mariadb-plugin-provider-lz4 mariadb-plugin-provider-lzma  
  mariadb-plugin-provider-lzo mariadb-plugin-provider-snappy  
  mariadb-server-core mysql-common perl perl-modules-5.36 pv rsync  
Suggested packages:  
  gawk-doc libnldb-perl libnet-daemon-perl libsql-statement-perl  
  libdata-dump-perl libipc-sharedcache-perl libbusiness-isbn-perl libwww-perl  
  mailx mariadb-test netcat-openbsd perl-doc libterm-readline-gnu-perl  
  | libterm-readline-perl-perl make libtap-harness-archive-perl doc-base  
  python3-braceexpand  
The following NEW packages will be installed:
```

Figura 1: Instalación de MariaDB server.

A continuación ejecutamos el siguiente comando, para asegurar el SGBD. En realidad no es un comando, es un script especial que permite el aumento de la seguridad de las bases de datos de MariaDB o MySQL.

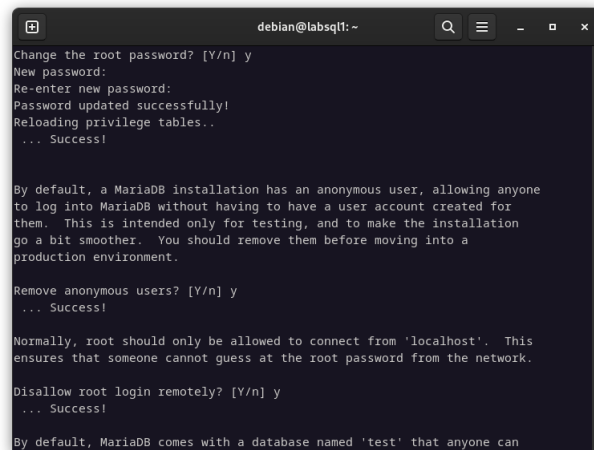
```
1 sudo mysql_secure_installation
```



```
debian@labsql1: ~  
$ sudo mysql_secure_installation  
  
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB  
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!  
  
In order to log into MariaDB to secure it, we'll need the current  
password for the root user. If you've just installed MariaDB, and  
haven't set the root password yet, you should just press enter here.  
  
Enter current password for root (enter for none):  
OK, successfully used password, moving on...  
  
Setting the root password or using the unix_socket ensures that nobody  
can log into the MariaDB root user without the proper authorisation.  
  
You already have your root account protected, so you can safely answer 'n'.  
  
Switch to unix_socket authentication [Y/n] y  
Enabled successfully!  
Reloading privilege tables..  
... Success!  
  
You already have your root account protected, so you can safely answer 'n'.
```

Figura 2: Ejecución de mysql_secure_installation.

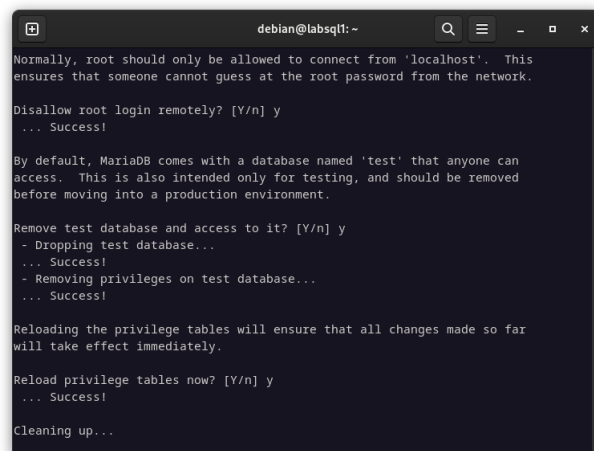
En la captura anterior, permitimos la conexión del tipo solo con Sockets, es decir, que



```
debian@labsql1: ~  
Change the root password? [Y/n] y  
New password:  
Re-enter new password:  
Password updated successfully!  
Reloading privilege tables..  
... Success!  
  
By default, a MariaDB installation has an anonymous user, allowing anyone  
to log into MariaDB without having to have a user account created for  
them. This is intended only for testing, and to make the installation  
go a bit smoother. You should remove them before moving into a  
production environment.  
  
Remove anonymous users? [Y/n] y  
... Success!  
  
Normally, root should only be allowed to connect from 'localhost'. This  
ensures that someone cannot guess at the root password from the network.  
  
Disallow root login remotely? [Y/n] y  
... Success!  
  
By default, MariaDB comes with a database named 'test' that anyone can
```

Figura 3: Ejecución de mysql_secure_installation - Parte de Usuarios y login.

En la captura anterior, eliminamos las tablas privilegiadas, eliminamos los usuarios anónimos que permiten que sean autenticados hacia el SGBD sin tener un usuario en concreto creado por el administrador de bases de datos. Por último bloqueamos la autenticación de root de manera remota.



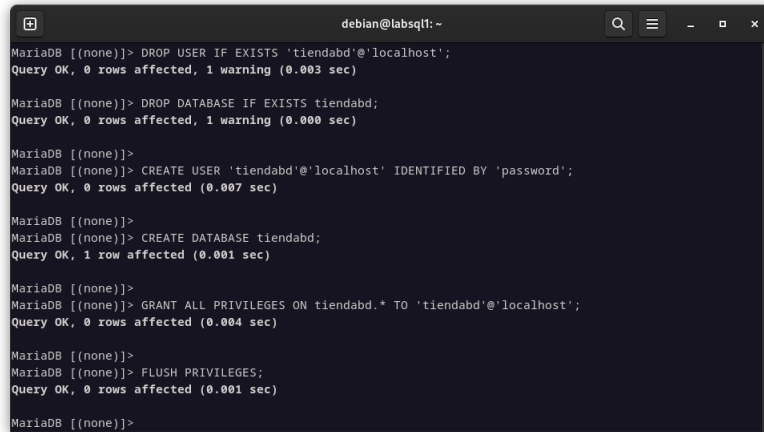
```
debian@labsql1: ~  
Normally, root should only be allowed to connect from 'localhost'. This  
ensures that someone cannot guess at the root password from the network.  
  
Disallow root login remotely? [Y/n] y  
... Success!  
  
By default, MariaDB comes with a database named 'test' that anyone can  
access. This is also intended only for testing, and should be removed  
before moving into a production environment.  
  
Remove test database and access to it? [Y/n] y  
- Dropping test database...  
... Success!  
- Removing privileges on test database...  
... Success!  
  
Reloading the privilege tables will ensure that all changes made so far  
will take effect immediately.  
  
Reload privilege tables now? [Y/n] y  
... Success!  
  
Cleaning up...
```

Figura 4: Ejecución de mysql_secure_installation - Eliminación de base de datos de test y su acceso.

Ahora tenemos que borrar las bases de datos de test, así como recargar los permisos de acceso.

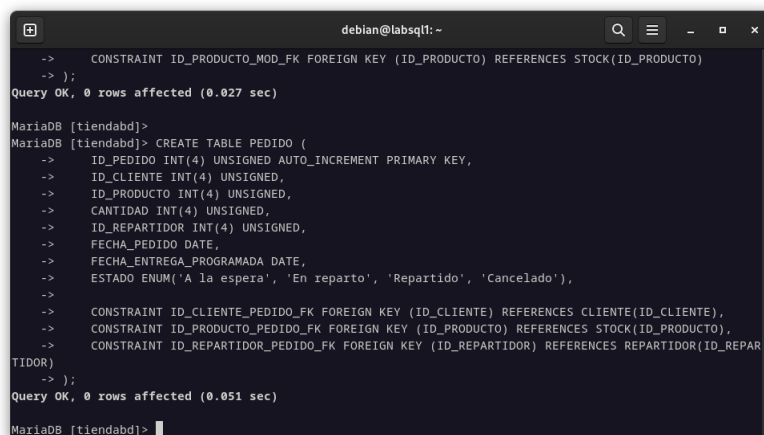
2. Datos de la base de datos

Ahora los datos de la base de datos que son introducidos, de manera general son los siguientes:



```
debian@labsql1:~  
MariaDB [(none)]> DROP USER IF EXISTS 'tiendabd'@'localhost';  
Query OK, 0 rows affected, 1 warning (0.003 sec)  
  
MariaDB [(none)]> DROP DATABASE IF EXISTS tiendabd;  
Query OK, 0 rows affected, 1 warning (0.000 sec)  
  
MariaDB [(none)]>  
MariaDB [(none)]> CREATE USER 'tiendabd'@'localhost' IDENTIFIED BY 'password';  
Query OK, 0 rows affected (0.007 sec)  
  
MariaDB [(none)]>  
MariaDB [(none)]> CREATE DATABASE tiendabd;  
Query OK, 1 row affected (0.001 sec)  
  
MariaDB [(none)]>  
MariaDB [(none)]> GRANT ALL PRIVILEGES ON tiendabd.* TO 'tiendabd'@'localhost';  
Query OK, 0 rows affected (0.004 sec)  
  
MariaDB [(none)]>  
MariaDB [(none)]> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.001 sec)  
  
MariaDB [(none)]>
```

Figura 5: Ejecución de SQL - Creando usuario, base de datos y permisos.



```
debian@labsql1:~  
-> CONSTRAINT ID_PRODUCTO_MOD_FK FOREIGN KEY (ID_PRODUCTO) REFERENCES STOCK(ID_PRODUCTO)  
-> );  
Query OK, 0 rows affected (0.027 sec)  
  
MariaDB [tiendabd]>  
MariaDB [tiendabd]> CREATE TABLE PEDIDO (  
-> ID_PEDIDO INT(4) UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
-> ID_CLIENTE INT(4) UNSIGNED,  
-> ID_PRODUCTO INT(4) UNSIGNED,  
-> CANTIDAD INT(4) UNSIGNED,  
-> ID_REPARTIDOR INT(4) UNSIGNED,  
-> FECHA_PEDIDO DATE,  
-> FECHA_ENTREGA_PROGRAMADA DATE,  
-> ESTADO ENUM('A la espera', 'En reparto', 'Repartido', 'Cancelado'),  
->  
-> CONSTRAINT ID_CLIENTE_PEDIDO_FK FOREIGN KEY (ID_CLIENTE) REFERENCES CLIENTE(ID_CLIENTE),  
-> CONSTRAINT ID_PRODUCTO_PEDIDO_FK FOREIGN KEY (ID_PRODUCTO) REFERENCES STOCK(ID_PRODUCTO),  
-> CONSTRAINT ID_REPARTIDOR_PEDIDO_FK FOREIGN KEY (ID_REPARTIDOR) REFERENCES REPARTIDOR(ID_REPAR  
TIDOR)  
-> );  
Query OK, 0 rows affected (0.051 sec)  
  
MariaDB [tiendabd]>
```

Figura 6: Ejecución de SQL - Creando tablas dentro de la base de datos.

```
debian@labsql1:~  
MariaDB [tiendabd]> INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Jersey XXL', 10);  
Query OK, 1 row affected (0.002 sec)  
  
MariaDB [tiendabd]> INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Patatas L', 10);  
Query OK, 1 row affected (0.003 sec)  
  
MariaDB [tiendabd]> INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Patatas M', 10);  
Query OK, 1 row affected (0.002 sec)  
  
MariaDB [tiendabd]> INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Patatas XL', 10);  
Query OK, 1 row affected (0.003 sec)  
  
MariaDB [tiendabd]> INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Patatas XXL', 10);  
Query OK, 1 row affected (0.002 sec)  
  
MariaDB [tiendabd]>  
MariaDB [tiendabd]> INSERT INTO REPARTIDOR (NOMBRE, PASSWORD) VALUES ('repartidor', '1234');  
Query OK, 1 row affected (0.003 sec)  
  
MariaDB [tiendabd]>  
MariaDB [tiendabd]> --INSERT INTO PEDIDO (ID_CLIENTE, ID_PRODUCTO, ID_REPARTIDOR, FECHA_PEDIDO, FECHA_ENT  
REGA_PROGRAMADA, REPARTIDO, EN_REPARTO) VALUES (1,1,1,NOW(),NOW(),0,0);  
MariaDB [tiendabd]>
```

Figura 7: Ejecución de SQL - Insertando datos en tablas dentro de la base de datos..

3. Script de Bash para CRON

El script realiza las copias de seguridad periódicas con mysqldump, con algunos parámetros ajustables que son explicados dentro del mismo, en resumen se permite afinar el dump según los parámetros que exiga el administrador. Así como indicar a través de variables los datos de la base de datos y su usuario.

El usuario debe tener los permisos indicados dentro del script, de lo contrario no podrá hacer el dump.

```
1  #!/bin/bash
2  # AUTOR: Pedro Antonio Mayorgas Parejo
3  # Fecha 20 de Agosto de 2023
4  ACTUALUID=$(id -u)
5  if [ $ACTUALUID != '0' ]; then
6      echo "Error: root UID Not found, please use sudo" >&2
7      exit 127
8  fi
9
10 # THIS SCRIPT DO LOGICAL BACKUPS DUMPING TO A .SQL A COMPLETE DATABASE
11 # PLEASE DO NOT INCLUDE LAST /
12 backup_subdir=/var/backups/sql
13
14 if [ ! -d $backup_subdir ]; then
15     echo "Backup dir not found!!" >&2
16     mkdir -p $backup_subdir
17 fi
18
19 # The database user require for executing this script
20 # SELECT PRIVILEGE FOR DUMPING TABLES
21 # VIEW PRIVILEGE FOR DUMPING VIEWS
22 # TRIGGER PRIVILEGE FOR DUMPING TRIGGERS
23 # LOCK TABLES FOR DUMPING WITH ARG --single-transaction and --add-locks
24 # PROCESS FOR --no-tablespaces
25 db_user="tiendabd"
26 db_userpass="password"
27 db_conn="localhost"
28 db_name="tiendabd"
29 # BOOLEANS: TRUE OR FALSE
30 # BOOLEAN: SET TRUE IF YOU NEED DUMP FOR ALL DB. FALSE IF YOU NEED DUMP OF
31 # BOOLEAN: SET LOCKING DB FOR CREATING DUMPING WITH RACE CONDITION OF
    PARALLEL OR CONCURRENT TRANSACTIONS MISSED ARGS --single-transaction --
    add-locks
32 lock_db="TRUE"
33 # BOOLEAN: SET IF YOU NEED DROP SENTENCE BEFORE CREATE DATABASE OR CREATE
    TABLE
34 add_drop="TRUE"
35 # BOOLEAN: SET IF YOU NEED REPLACE DATA ON TABLES INSTEAD OF INSERTING
36 replace_sentences="TRUE"
37 # BOOLEAN: SET IF YOU USE SOCKET OR FALSE FOR USE IP STACK AND db_conn
38 use_socket="TRUE"
39
40 # WE ARE USING A ARRAY FOR APPENDING THE PARAMETERS TO THE BASE COMMAND
```

```

41 declare -a DUMPCOMMAND=("mysqldump --user=${db_user} --password=${
    db_userpass} --databases ${db_name}")
42
43 if [ $add_drop = 'TRUE' ]; then
44     DUMPCOMMAND+=(' --add-drop-database --add-drop-table ')
45 fi
46
47 if [ $lock_db = 'TRUE' ]; then
48     DUMPCOMMAND+=(' --add_locks ')
49 fi
50
51 if [ $replace_sentences = 'TRUE' ]; then
52     DUMPCOMMAND+=(' --replace ')
53 fi
54
55 if [ $use_socket = 'TRUE' ]; then
56     DUMPCOMMAND+=(" --host=${db_conn}")
57 fi
58
59 ACTUALDATE='date +%y-%m-%d-%H:%M'
60
61 DUMPCOMMAND+=(" --result-file=${backup_subdir}/dump-${ACTUALDATE}.sql")
62
63 # echo "${DUMPCOMMAND[@]}"
64 ${DUMPCOMMAND[@]}

```


Luego el siguiente script, funciona como archivador, para que no se generen infinitos archivos .sql, permitiendo que se puedan almacenar por cada día todos los .sql generados y poder limpiar la carpeta para el día siguiente.

```

1  #!/bin/bash
2  # AUTOR: Pedro Antonio Mayorgas Parejo
3  # Fecha 20 de Agosto de 2023
4  ACTUALUID=$(id -u)
5  if [ $ACTUALUID != '0' ]; then
6      echo "Error: root UID Not found, please use sudo" >&2
7      exit 127
8  fi
9
10 # THIS SCRIPT DO ARCHIVAL .SQL ON A SUBFOLDER FOR CLEANING THE BACKUP
    SUBDIR
11 # PLEASE DO NOT INCLUDE LAST /
12 backup_subdir=/var/backups/sql
13 backup_archival_subdir=/var/backups/archival
14
15 if [ ! -d $backup_subdir ]; then
16     echo "Backup dir not found!!"
17     mkdir -p $backup_subdir
18 fi
19 if [ ! -d $backup_archival_subdir ]; then
20     echo "Backup archival not found!!" >&2
21     mkdir -p $backup_archival_subdir
22 fi
23
24 # JUMPING TO THE FOLDER
25 cd ${backup_subdir}
26
27 # GETTING THE ACTUAL DAY
28 ACTUALDATE=$(date +%Y-%m-%d)
29
30 # CREATING TAR AND MOVING TO ARCHIVAL FOLDER
31 tar -czf dumps-${ACTUALDATE}.tar.gz $(ls | grep ".sql") && mv dumps-${
    ACTUALDATE}.tar.gz ${backup_archival_subdir}
32
33 # REMOVING .SQL
34 rm $(ls | grep .sql)

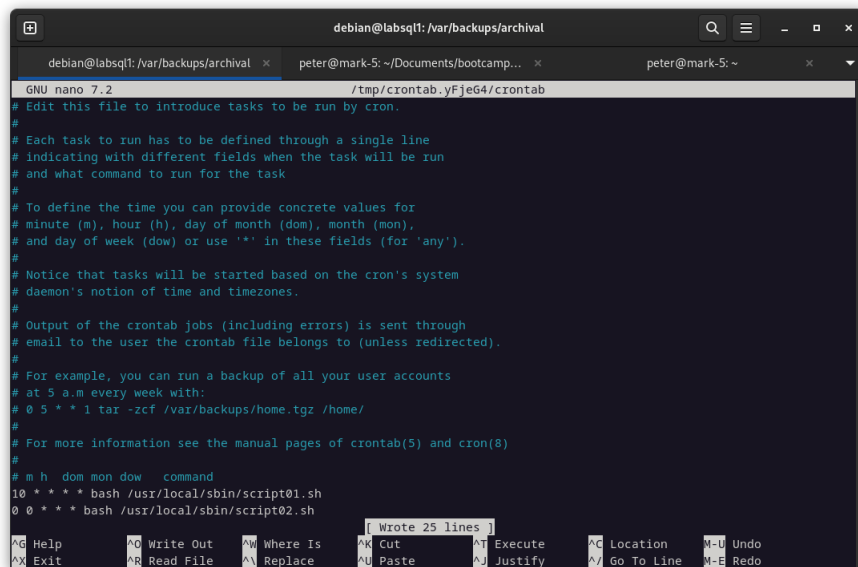
```

Ahora movemos los scripts a un directorio más adecuado para la ejecución por parte de cron.

```

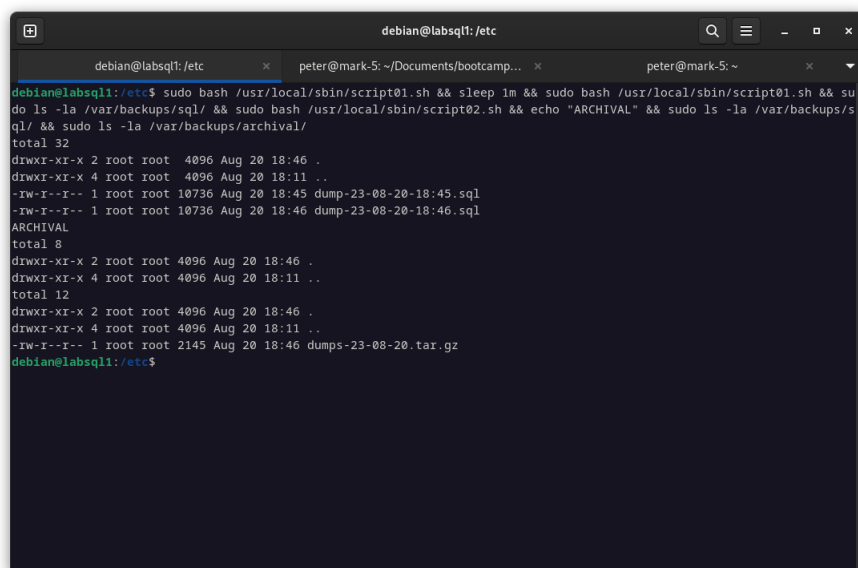
1  cp script01.sh /usr/local/sbin
2  cp script02.sh /usr/local/sbin
3  sudo apt install cron
4  # EDITAMOS EL CRONTAB
5  sudo crontab -e

```



```
debian@labsql1: /var/backups/archival
GNU nano 7.2 /tmp/crontab.yFjeG4/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
10 * * * * bash /usr/local/sbin/script01.sh
0 0 * * * bash /usr/local/sbin/script02.sh
```

Figura 8: Crontab editado



```
debian@labsql1: /etc
debian@labsql1:/etc$ sudo bash /usr/local/sbin/script01.sh && sleep 1m && sudo bash /usr/local/sbin/script01.sh && su
do ls -la /var/backups/sql/ && sudo bash /usr/local/sbin/script02.sh && echo "ARCHIVAL" && sudo ls -la /var/backups/s
ql/ && sudo ls -la /var/backups/archival/
total 32
drwxr-xr-x 2 root root 4096 Aug 20 18:46 .
drwxr-xr-x 4 root root 4096 Aug 20 18:11 ..
-rw-r--r-- 1 root root 10736 Aug 20 18:45 dump-23-08-20-18:45.sql
-rw-r--r-- 1 root root 10736 Aug 20 18:46 dump-23-08-20-18:46.sql
ARCHIVAL
total 8
drwxr-xr-x 2 root root 4096 Aug 20 18:46 .
drwxr-xr-x 4 root root 4096 Aug 20 18:11 ..
total 12
drwxr-xr-x 2 root root 4096 Aug 20 18:46 .
drwxr-xr-x 4 root root 4096 Aug 20 18:11 ..
-rw-r--r-- 1 root root 2145 Aug 20 18:46 dumps-23-08-20.tar.gz
debian@labsql1:/etc$
```

Figura 9: Resultado si ejecutamos los scripts