Bash Scripts

Mysql dump Script

Create a script to dump MySQL db

1.To create a .sh file

```
nano mysql_script.sh
```

To add the script for mysql dump

```
#!/bin/bash
DB_USER="sasi-user"
DB_PASSWORD="Sasikumar#123"
DB_NAME="mydb_mysql"

OUTPUT_DIR="/opt/mysql_backup1"

TIMESTAMP=$(date +%Y%m%d%H%M%S)

mkdir -p "$OUTPUT_DIR"

DUMP_FILE="$OUTPUT_DIR/$DB_NAME-$TIMESTAMP.sql"

mysqldump --skip-add-drop-table -u"$DB_USER" -p"$DB_PASSWORD" "$DB_NAME" > "$DUMP_FILE"
if [ $? -eq 0 ]; then
  echo "Database dump successful. File: $DUMP_FILE"
else
  echo "Error: Database dump failed."
fi
```

```
  GNU nano 6.2                                                    mysql_script.sh
#!/bin/bash

DB_USER="sasi-user"
DB_PASSWORD="Sasikumar#123"
DB_NAME="mydb_mysql"

OUTPUT_DIR="/opt/mysql_backup1"

TIMESTAMP=$(date +%Y%m%d%H%M%S)

mkdir -p "$OUTPUT_DIR"

DUMP_FILE="$OUTPUT_DIR/$DB_NAME-$TIMESTAMP.sql"

mysqldump --skip-add-drop-table -u"$DB_USER" -p"$DB_PASSWORD" "$DB_NAME" > "$DUMP_FILE"

if [ $? -eq 0 ]; then
  echo "Database dump successful. File: $DUMP_FILE"
else
  echo "Error: Database dump failed."
fi
```

2.To change the permission

```
chmod u+x mysql_script.sh
```

```
root@ip-172-31-5-149:/home/ubuntu/mysql# nano mysql_script.sh
root@ip-172-31-5-149:/home/ubuntu/mysql# chmod u+x mysql_script.sh
```

3.To run the bash script

```
./mysql_script.sh
```

```
root@ip-172-31-5-149:/home/ubuntu/mysql# ./mysql_script.sh
mysqldump: [Warning] Using a password on the command line interface can be insecure.
Database dump successful. File: /opt/mysql_backup1/mydb_mysql-20231206081119.sql
root@ip-172-31-5-149:/home/ubuntu/mysql#
```

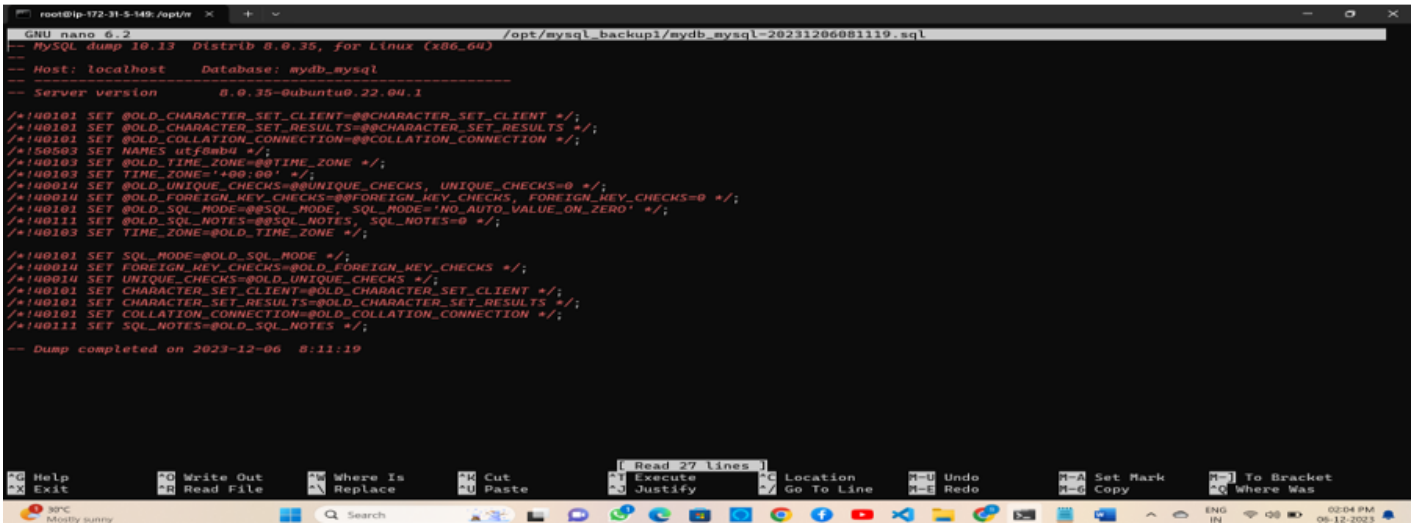
4.To check the file will be completed

```
tail -f mydb_mysql-20231206081119.sql
```

```
root@ip-172-31-5-149:/opt/mysql_backup1# tail -f  /opt/mysql_backup1/mydb_mysql-20231206081119.sql

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2023-12-06  8:11:19
```

5.Using nano to check the file

```
nano /opt/mysql_backup1/mydb_mysql-20231206081119.sql
```



6.Now making changes in database

```
mysql> use mydb_mysql;
Database changed
mysql> create table mytable (
    -> id int primary key,
    -> name varchar(255),
    -> email varchar(255)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> insert into mytable values (1, "sasi", "sasi1@gmail.com");
Query OK, 1 row affected (0.02 sec)

mysql> select * from mytable
    -> ;
+----+------+-----------------+
| id | name | email           |
+----+------+-----------------+
|  1 | sasi | sasi1@gmail.com |
+----+------+-----------------+
1 row in set (0.00 sec)

mysql> exit
Bye
```

7.Now again run the scripts

```
./mysql_script.sh
```

```
root@ip-172-31-5-149:/home/ubuntu/mysql# ./mysql_script.sh
mysqldump: [Warning] Using a password on the command line interface can be insecure.
Database dump successful. File: /opt/mysql_backup1/mydb_mysql-20231206082319.sql
```

8.To check the dump file using nano editor

```
nano /opt/mysql_backup1/mydb_mysql-20231206082319.sql
```

```sql
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `mytable` (
  `id` int NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `mytable`
--

LOCK TABLES `mytable` WRITE;
/*!40000 ALTER TABLE `mytable` DISABLE KEYS */;
INSERT INTO `mytable` VALUES (1,'sasi','sasi1@gmail.com');
/*!40000 ALTER TABLE `mytable` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
```

PSQL Dump Script

POSTGRESQL Create a script to dump POSTGRESQL Database

1.To create a database for postgresql

```
su postgres
psql
```

```
create database mypostgres;
```

2.To create user with grant privileges for this database

```
CREATE USER "user" WITH PASSWORD 'your_passwd';
```

```
postgres=# CREATE USER "sasi-user" WITH PASSWORD 'Sasikumar#123';
CREATE ROLE
postgres=# ALTER USER "sasi-user" mypostgres;
ERROR:  unrecognized role option "mypostgres"
LINE 1: ALTER USER "sasi-user" mypostgres;
                               ^
postgres=# ALTER USER "sasi-user" CREATEDB;
ALTER ROLE
postgres=# GRANT ALL PRIVILEGES ON DATABASE mypostgres TO "sasi-user";
GRANT
```

3.To add the user in postgresql file

```
nano /etc/postgresql/14/main/pg_hba.conf
```

3.To add the user in conf file

```
# TYPE  DATABASE        USER            ADDRESS            METHOD
local   mypostgres      user                                md5
```

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
local   mypostgres      sasi-user                                md5

# "local" is for Unix domain socket connections only
```

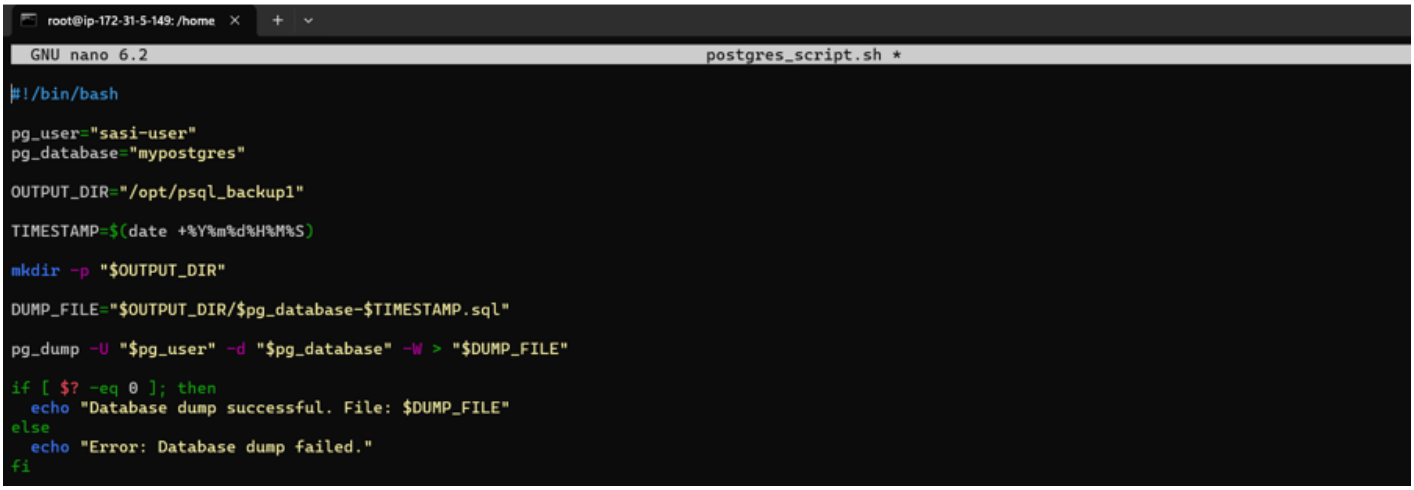
4.Now create a .sh script file for postgresql

Go to your script folder and create file

```
nano postgres_script.sh
```

5.To add the script content for postgresql

```
#! /bin/bash

pg_user="sasi-user"
pg_database="mypostgres"

OUTPUT_DIR="/opt/psql_backup1"

TIMESTAMP=$(date +%Y%m%d%H%M%S)

mkdir -p "$OUTPUT_DIR"

DUMP_FILE="$OUTPUT_DIR/$pg_database-$TIMESTAMP.sql"

pg_dump -U "$pg_user" -d "$pg_database" -W > "$DUMP_FILE"

if [ $? -eq 0 ]; then
  echo "Database dump successful. File: $DUMP_FILE"
else
  echo "Error: Database dump failed."
fi
```



6.To run the bash script

```
./postgres_script.sh
```

```
root@ip-172-31-5-149:/home/ubuntu/postgres# ./postgres_script.sh
Password:
Database dump successful. File: /opt/psql_backup1/mypostgres-20231206094318.sql
```

7.To check the dump file

```
nano /opt/psql_backup1/mypostgres-20231206094318.sql
```

```
  GNU nano 6.2                                    /opt/psql_backup1/mypostgres-20231206094318.sql
--
-- PostgreSQL database dump
--

-- Dumped from database version 14.9 (Ubuntu 14.9-0ubuntu0.22.04.1)
-- Dumped by pg_dump version 14.9 (Ubuntu 14.9-0ubuntu0.22.04.1)

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;

--
```

8.Now make the changes in db

Add the table and insert the values

```
postgres=# use mypostgres
postgres-# ;
ERROR:  syntax error at or near "use"
LINE 1: use mypostgres
        ^
postgres=# \c
You are now connected to database "postgres" as user "postgres".
postgres=# \c mypostgres;
You are now connected to database "mypostgres" as user "postgres".
mypostgres=# create table mytable (
mypostgres(# id int primary key,
mypostgres(# name varchar(255),
mypostgres(# mail varchar(255)
mypostgres(# );
CREATE TABLE
```

```
mypostgres=# INSERT INTO mytable VALUES (1, 'NEwuser', 'new@gmail.com');
INSERT 0 1
mypostgres=# \q
postgres@ip-172-31-5-149:/home/ubuntu/postgres$
```

9.To grant the select permissions

```
GRANT SELECT ON TABLE mytable TO "sasi-user";
```

10.After making the changes, then the script

```
./postgres_script.sh
```

11.Now check the created dump file with nano editor

```
nano /opt/psql_backup1/mypostgres-20231206100711.sql
```

```
--

CREATE TABLE public.mytable (
    id integer NOT NULL,
    name character varying(255),
    mail character varying(255)
);


ALTER TABLE public.mytable OWNER TO postgres;

--
-- Data for Name: mytable; Type: TABLE DATA; Schema: public; Owner: postgres
--

COPY public.mytable (id, name, mail) FROM stdin;
1       NEwuser new@gmail.com
\.
```

Script for to delete .gz files

Script for to delete .gz files

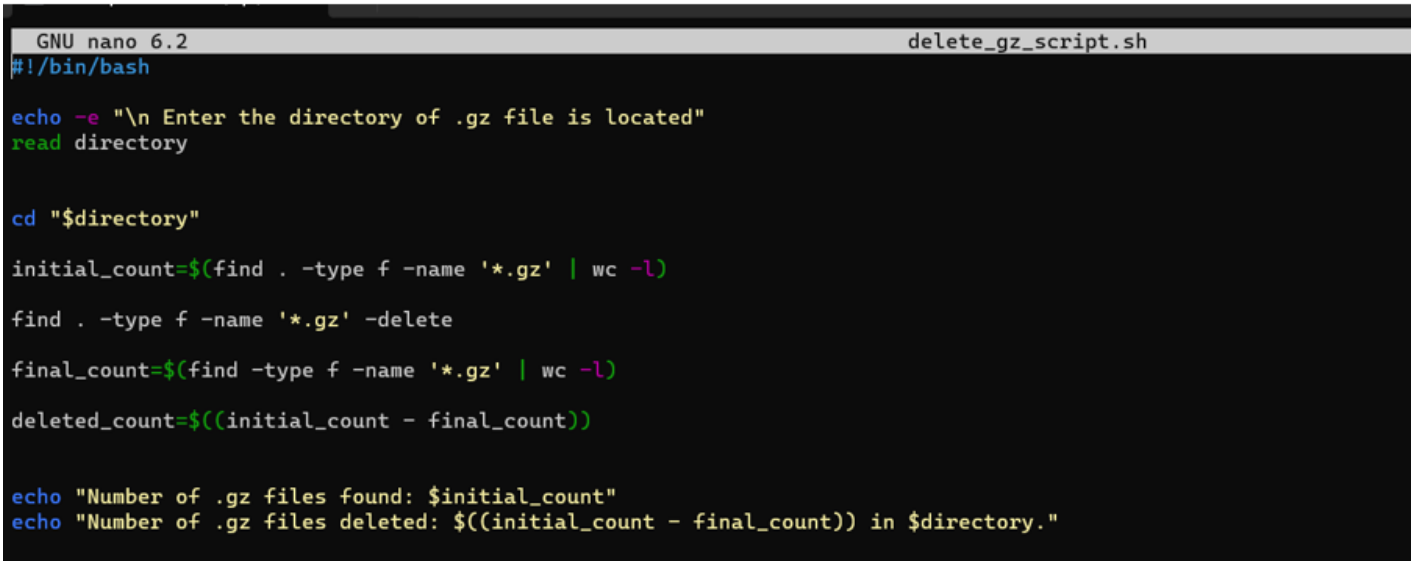
1.To create a script file for deleting the .gz files

Go to the your script folder and create file

```
nano delete_gz_script.sh
```

2.Add the script for deleted the .gz files

```
#! /bin/bash
echo -e "\n Enter the directory of .gz file is located"
read directory
cd "$directory"
initial_count=$(find . -type f -name '*.gz' | wc -l)
find . -type f -name '*.gz' -delete
final_count=$(find -type f -name '*.gz' | wc -l)
deleted_count=$((initial_count - final_count))
echo "Number of .gz files found: $initial_count"
echo "Number of .gz files deleted: $((initial_count - final_count)) in $directory."
```



3.To give the permissions for the file

```
chmod u+x delete_gz_script.sh
```

```
root@ip-172-31-5-149:/opt/newtype# nano delete_gz_script.sh
root@ip-172-31-5-149:/opt/newtype# chmod u+x delete_gz_script.sh
root@ip-172-31-5-149:/opt/newtype#
```

4.To create a sample .gz files

```
touch file1.gz file2.gz file3.gz file3.gz
```

```
root@ip-172-31-5-149:/opt/newtype# touch file1.gz file2.gz file3.gz file3.gz
root@ip-172-31-5-149:/opt/newtype# ls
delete_gz_script.sh  file1.gz  file2.gz  file3.gz  test2.sh
root@ip-172-31-5-149:/opt/newtype#
```

5.After creating the sample .gz file, now run the script

```
./delete_gz_script.sh
```

```
root@ip-172-31-5-149:/opt/newtype# ./delete_gz_script.sh

 Enter the directory of .gz file is located

Number of .gz files found: 3
Number of .gz files deleted: 3 in .
root@ip-172-31-5-149:/opt/newtype# ls
delete_gz_script.sh  test2.sh
root@ip-172-31-5-149:/opt/newtype#
```

Find recently edited file

Find the recently edited file

To create a script file for recently edited files

```
nano find_recent.sh
```

2.To add the script content

```
#!/bin/bash
recent_files=$(find / -type f -printf "%T@ %p\n" 2>/dev/null | sort -n | tail -n 5 | cut -d '
' -f 2)
file_count=$(echo "$recent_files" | wc -l)
echo "Number of files edited in recent: $file_count"
if [ "$file_count" -gt 0 ]; then
    echo "Edited file directory paths:"
    echo "$recent_files"
else
    echo "No files found in the entire file system."
fi
```

2>/dev/null: Redirects error messages (stderr) to /dev/null to suppress permission-denied messages

sort -n: Pipes the output of the find command to sort -n, which sorts the lines numerically based on the modification time.

tail -n 5: Pipes the sorted output to tail -n 5, which extracts the last 5 lines (i.e., the 5 most recently modified files).

cut -d ' ' -f 2: Pipes the output to cut -d ' ' -f 2, which uses a space (' ') as the delimiter (-d) and extracts the second field (-f 2).

```
root@ip-172-31-5-149: /opt/n
  GNU nano 6.2                                                    find_recent.sh
#!/bin/bash

# Use find to locate files across the entire file system and sort them by modification time
recent_files=$(find / -type f -printf "%T@ %p\n" 2>/dev/null | sort -n | tail -n 5 | cut -d ' ' -f 2)

# Count the number of recently edited files
file_count=$(echo "$recent_files" | wc -l)

echo "Number of files edited in recent: $file_count"

if [ "$file_count" -gt 0 ]; then
    echo "Edited file directory paths:"
    echo "$recent_files"
else
    echo "No files found in the entire file system."
fi
```

3.To change the permission for scripted file

```
chmod +x find_recent.sh
```

```
root@ip-172-31-5-149:/opt/newtype# nano find_recent.sh
root@ip-172-31-5-149:/opt/newtype# chmod +x find_recent.sh
```

4.Now run the script

```
./find_recent.sh
```

```
root@ip-172-31-5-149:/opt/newtype# ./find_recent.sh
Number of files edited in recent: 5
Edited file directory paths:
/proc/7516/task/7516/net/wireless
/proc/7516/task/7516/net/xfrm_stat
/run/postgresql/14-main.pg_stat_tmp/db_0.stat
/run/postgresql/14-main.pg_stat_tmp/db_13761.stat
/run/postgresql/14-main.pg_stat_tmp/global.stat
```

Scripts to run python commands

Scripts to run python commands

1.To create a .sh script file for run python commands

```
nano python_script.sh
```

2.To add the script in the .sh file

```
#! /bin/bash
echo "Enter your Python code (press Ctrl+D to finish input):"
python_code=$(cat)
# Run the entered Python code and capture the output
output=$(python3 -c "$python_code" 2>&1)
echo "Output of code:"
echo "$output"
```

```
  GNU nano 6.2                                        python_script.sh
#!/bin/bash

echo "Enter your Python code (press Ctrl+D to finish input):"
python_code=$(cat)

# Run the entered Python code and capture the output
output=$(python3 -c "$python_code" 2>&1)

echo "Output of code:"
echo "$output"
```

3.Save and exit the file

4.To run the python script

```
./python_script.sh
```

5.To enter the python and press ctrl+D and see the output

```
root@ip-172-31-5-149:/opt/python# ./python_script.sh
Enter your Python code (press Ctrl+D to finish input):
a = 10
b = 20
print(a+b)
Output of code:
30
```

Script to create docker image and create docker container

Script to create docker image and create docker container

1.To create a .sh script file for run docker container

```
nano docker_cont.sh
```

2.To add the scripts for run docker container

```
#!/bin/bash

apt install docker.io -y
read -p "Enter IMAGE_NAME: " IMAGE_NAME
read -p "Enter CONTAINER_NAME: " CONTAINER_NAME
read -p "Enter PORT_NUM: " PORT_NUM

# Check if docker command is available
if ! command -v docker &> /dev/null; then
    echo "Error: Docker not found. Please install Docker before running this script."
    exit 1
fi

# Build Docker image
docker build -t $IMAGE_NAME .

# Create and run Docker container
docker run -d --name $CONTAINER_NAME -p $PORT_NUM $IMAGE_NAME

# Display container information
```

```
docker ps -a |  grep $CONTAINER_NAME


#To execute the container

 docker exec -it $CONTAINER_NAME /bin/bash
```

```
  GNU nano 6.2                                        docker_cont.sh
#!/bin/bash

apt install docker.io -y
read -p "Enter IMAGE_NAME: " IMAGE_NAME
read -p "Enter CONTAINER_NAME: " CONTAINER_NAME
read -p "Enter PORT_NUM: " PORT_NUM

# Check if docker command is available
if ! command -v docker &> /dev/null; then
    echo "Error: Docker not found. Please install Docker before running this script."
    exit 1
fi

# Build Docker image
docker build -t $IMAGE_NAME .

# Create and run Docker container
docker run -d --name $CONTAINER_NAME -p $PORT_NUM $IMAGE_NAME

# Display container information
docker ps -a | grep $CONTAINER_NAME

 docker exec -it $CONTAINER_NAME /bin/bash
```

3.To run the docker script

```
./docker_cont.sh
```

4.Enter the docker details and see the output

For example:

Image name : nginx Container name : mycont1 Port number : 83:80

See the output

```
root@ip-172-31-5-149:/opt/docker# ./docker_cont.sh
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker.io is already the newest version (24.0.5-0ubuntu1~22.04.1).
0 upgraded, 0 newly installed, 0 to remove and 33 not upgraded.
Enter IMAGE_NAME: nginx
Enter CONTAINER_NAME: mycont1
Enter PORT_NUM: 83:80
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

unable to prepare context: unable to evaluate symlinks in Dockerfile path: lstat /opt/docker/Dockerfile: no such file or
 directory
2bf7131e54d0bf13e133bc1c64cb8fbca375d80aa3e975a6fc792d370ea99e71
2bf7131e54d0   nginx     "/docker-entrypoint.…"   1 second ago    Up Less than a second   0.0.0.0:83->80/tcp, :::83->80/
tcp   mycont1
root@2bf7131e54d0:/#
```

5.Now test the container in browser