EXPERIMENT 2

AIM: Working with Docker (Basic Commands)

1. Search a Redis Image on Docker Hub :

```
docker pull redis
```

docker pull redis : downloads the Redis image from Docker-hub to your local machine. Docker-hub is a repository where Docker images are stored. The Redis image is used to run a Redis server. By pulling the image, you're preparing your system to create containers based on this image.

```
C:\Users\aksha>docker pull redis
Using default tag: latest
latest: Pulling from library/redis
e4fff0779e6d: Pull complete
d1dde3db2ec5: Pull complete
1d321a003dde: Pull complete
d65aedb2f012: Pull complete
4018f93716a2: Pull complete
b0967b02e8cf: Pull complete
4f4fb700ef54: Pull complete
d288b86f5d06: Pull complete
Digest: sha256:878983f8f5045b28384fc300268cec62bca3b14d5e1a448bec21f28cfcc7bf78
Status: Downloaded newer image for redis:latest
docker.io/library/redis:latest

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview redis
```

2. Run Docker Container of Redis Image in Background :

```
docker run -it --name myredis redis:latest /bin/bash
```

· *docker run*: This command creates and starts a new container from an image.

· *-it* : The -i flag keeps the STDIN open, and the -t flag allocates a pseudo-TTY, allowing you to interact with the container via a terminal session.

· *--name myredis* : This names the container "*myredis*" for easy reference.

· redis:latest: Specifies the Redis image (using the latest version) to use for the container.

· */bin/bash* : This opens a Bash shell inside the container, allowing you to interact with it directly. While this command starts the container interactively.

```
C:\Users\aksha>docker run -it --name myredis redis:latest /bin/bash
root@e88857e500c0:/data#
```

3. Run Docker PS and Docker PS –a :

```
docker ps
```

docker ps : This lists all running Docker containers. It shows container IDs, names, and other details about active containers.

```
docker ps -a
```

docker ps -a : This lists all containers, including those that are stopped. This is useful to see all containers that exist on your system, not just the ones that are currently running.

```
C:\Users\aksha>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS      NAMES
e88857e500c0   redis:latest   "docker-entrypoint.s…"   59 seconds ago   Up 58 seconds   6379/tcp   myredis

C:\Users\aksha>docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS      NAMES
e88857e500c0   redis:latest   "docker-entrypoint.s…"   About a minute ago   Up About a minute   6379/tcp   myredis

C:\Users\aksha>
```

4. Run Docker Container and take its console :

```
docker run -it --name myredis redis:latest /bin/bash
```

The command to start the container (*docker run -it --name myredis redis:latest /bin/bash*) is the same as in step 2, creating an interactive terminal session inside the container.

```
root@e88857e500c0:/data# redis-server --version
```

redis-server --version : Once inside the container, this command checks the version of the Redis server installed in the container. It confirms that the Redis server is correctly installed and running within the container.

```
C:\Users\aksha>docker run -it --name myredis redis:latest /bin/bash
root@e88857e500c0:/data# redis-server --version
Redis server v=7.4.0 sha=00000000:0 malloc=jemalloc-5.3.0 bits=64 build=c9b63216ff6b6742
root@e88857e500c0:/data#
```

5. Create a Docker Volume and connect it :

```
docker volume create myredisdata
```

docker volume create myredisdata : This command creates a Docker volume named *myredisdata*. A Docker volume is a persistent storage area that can be shared between containers or accessed even after a container is deleted.

```
docker run -it --name Myredis --mount source=myredisdata,target=/data redis /bin/bash
```

--mount source=myredisdata,target=/data : This option mounts the *myredisdata* volume to the */data* directory inside the container. Any data written to */data* inside the container will be stored in the *myredisdata* volume, persisting even if the container is removed.

```
root@a0a3c177e155:/data# touch /data/testfile.txt
```

touch /data/testfile.txt : This creates a new file named *testfile.txt* in the */data* directory, demonstrating that the volume is correctly mounted.

```
root@a0a3c177e155:/data# ls
```

Ls : This lists the contents of the /data directory, showing that *testfile.txt* has been successfully created, confirming that the volume is connected and functioning.

```
C:\Users\aksha>docker volume create myredisdata
myredisdata

C:\Users\aksha>docker run -it --name Myredis --mount source=myredisdata,target=/data redis /bin/bash
root@a0a3c177e155:/data# touch /data/testfile.txt
root@a0a3c177e155:/data# ls
testfile.txt
root@a0a3c177e155:/data#
```