Hands-On Exercise for Day16 of DevOps Professional training.

1. Create a folder for your activity. Check for availability of docker service:

```
osgdev@TG-DevOps-OS004:~$ mkdir dockerlab
osgdev@TG-DevOps-OS004:~$ cd dockerlab/
osgdev@TG-DevOps-OS004:~/dockerlab$

osgdev@TG-DevOps-OS004:~/dockerlab$ service docker status
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; disabled; vendor
preset:
   Active: inactive (dead)
     Docs: https://docs.docker.com
```

Note: Normally docker service is kept in stopped state in Topgear machine due to technical reasons. Hence you need to start it.

---

2. Start docker service and check for its availability. For password use the same password you used to login into Topgear machine.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ service docker start
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===
Authentication is required to start 'docker.socket'.
Authenticating as: osgdev,,, (osgdev)
Password:
==== AUTHENTICATION COMPLETE ===
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===
Authentication is required to start 'docker.service'.
Authenticating as: osgdev,,, (osgdev)
Password:

osgdev@TG-DevOps-OS004:~/dockerlab$ service docker status
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; disabled; vendor preset:
   Active: active (running) since Tue 2018-04-10 19:09:18 IST; 58s ago
     Docs: https://docs.docker.com
 Main PID: 12229 (dockerd)
    Tasks: 17
   Memory: 96.8M
      CPU: 590ms
   CGroup: /system.slice/docker.service
           ├─12229 /usr/bin/dockerd -H fd://
           └─12261 docker-containerd --config /var/run/docker/containerd/contain

Apr 10 19:09:16 TG-DevOps-OS004 dockerd[12229]: time="2018-04-10T19:09:16.844263
Apr 10 19:09:16 TG-DevOps-OS004 dockerd[12229]: time="2018-04-10T19:09:16.844314
Apr 10 19:09:16 TG-DevOps-OS004 dockerd[12229]: time="2018-04-10T19:09:16.844324
Apr 10 19:09:16 TG-DevOps-OS004 dockerd[12229]: time="2018-04-10T19:09:16.845790
Apr 10 19:09:17 TG-DevOps-OS004 dockerd[12229]: time="2018-04-10T19:09:17.615721
Apr 10 19:09:17 TG-DevOps-OS004 dockerd[12229]: time="2018-04-10T19:09:17.672155
Apr 10 19:09:18 TG-DevOps-OS004 dockerd[12229]: time="2018-04-10T19:09:18.072070
Apr 10 19:09:18 TG-DevOps-OS004 dockerd[12229]: time="2018-04-10T19:09:18.075702
Apr 10 19:09:18 TG-DevOps-OS004 dockerd[12229]: time="2018-04-10T19:09:18.103994
Apr 10 19:09:18 TG-DevOps-OS004 systemd[1]: Started Docker Application Container

osgdev@TG-DevOps-OS004:~/dockerlab$ docker version
Client:
```

```
 Version:  17.12.1-ce
 API version:  1.35
 Go version:  go1.9.4
 Git commit:  7390fc6
 Built:  Tue Feb 27 22:17:40 2018
 OS/Arch:  linux/amd64

Server:
 Engine:
  Version:  17.12.1-ce
  API version:  1.35 (minimum version 1.12)
  Go version:  go1.9.4
  Git commit:  7390fc6
  Built:  Tue Feb 27 22:16:13 2018
  OS/Arch:  linux/amd64
  Experimental:  false
```

Another formal test of Docker working is the hello-world image provided by Docker itself.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working
correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker
Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which
runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which
sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/
```

---

3. List the docker images already available in the given machine.

Note: the output is edited, probably you may see lot more images.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker image ls
REPOSITORY                                  TAG                 IMAGE ID
CREATED             SIZE
tomcat                                      9                   4f88b8ccbcd0
2 months ago        714MB
tomcat                                      7                   d10641f583b3
2 months ago        456MB
```

```
ubuntu                                  16.04               2a4cca5ac898
2 months ago        111MB
hello-world                             latest              f2a91732366c
4 months ago        1.85kB
```

---

4. We shall use Ubuntu:16.04 image which is compatible for our Ubuntu:16.04 machine to see what is inside the container.

Note: -i flag is for interactive access to container in a new terminal (-t)

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker container run -it
ubuntu:16.04
root@301a4c3c816c:/# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
```

Note: we used linux list (ls) command to see the file system inside the container.
You may open another terminal windows and check the filesystem on the host machine.
Look for similarity in file system between host machine and file system within the container.

```
osgdev@TG-DevOps-OS004:~$ ls /
bin   home            lib64       opt   sbin  tmp      vmlinuz.old
boot  initrd.img      lost+found  proc  snap  usr
dev   initrd.img.old  media       root  srv   var
etc   lib             mnt         run   sys   vmlinuz
```

You can also see the running container on the host machine:

```
osgdev@TG-DevOps-OS004:~$ docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
301a4c3c816c        ubuntu:16.04        "/bin/bash"         4 minutes
ago       Up 4 minutes                            naughty_brown
```

---

5. Let us make some change to file system environment of the container. Let us create a folder and file.

```
root@301a4c3c816c:/# mkdir MASTER
root@301a4c3c816c:/# touch testfile
root@301a4c3c816c:/# ls
MASTER  boot  etc   lib    media  opt   root  sbin  sys       tmp  var
bin     dev   home  lib64  mnt    proc  run   srv   testfile  usr
root@301a4c3c816c:/# exit
exit
osgdev@TG-DevOps-OS004:~/dockerlab$
```

Note: After making the change to the container we exited from the container. Let us check the status of container. You can observe that the container is no longer running. Hence we used -a flag to list all containers, where the status is shown as Exited About a minute ago.

```
osgdev@TG-DevOps-OS004:~$ docker container ls
```

```
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES

osgdev@TG-DevOps-OS004:~$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS                          PORTS
NAMES
301a4c3c816c        ubuntu:16.04        "/bin/bash"              7
minutes ago       Exited (0) About a minute ago
naughty_brown
4f83d502a259        hello-world         "/hello"                 13
minutes ago      Exited (0) 13 minutes ago
pensive_hugle
```

---

6. We can start the Exited(Stopped) container and can access the same again.

Note: you may identify and work with a container either using "CONTAINER ID" (301a4c3c816c) or using NAMES (naughty_brown)

```
osgdev@TG-DevOps-OS004:~$ docker container start 301a4c3c816c
301a4c3c816c

osgdev@TG-DevOps-OS004:~$ docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
301a4c3c816c        ubuntu:16.04        "/bin/bash"         13 minutes
ago      Up 8 seconds                           naughty_brown

osgdev@TG-DevOps-OS004:~$ docker attach naughty_brown
root@301a4c3c816c:/#

root@301a4c3c816c:/# ls
MASTER  boot  etc   lib    media  opt   root  sbin  sys       tmp  var
bin     dev   home  lib64  mnt    proc  run   srv   testfile  usr
root@301a4c3c816c:/#
```

Note: You can observe that the content of the container is intact, while it is being stopped and started again. Check for folder MASTER and file testfile

---

7. You can also stop the container from the system, on which the container is running.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker container stop 301a4c3c816c
301a4c3c816c
```

As you execute the above command, you can watch on the other window, where you are interacting with the running container automatically execute an "exit" command and comes out of container.

```
root@301a4c3c816c:/# ls
MASTER  boot  etc   lib    media  opt   root  sbin  sys       tmp  var
bin     dev   home  lib64  mnt    proc  run   srv   testfile  usr
root@301a4c3c816c:/# exit
```

```
osgdev@TG-DevOps-OS004:~$
```

Note: You can start and stop the container any number of times, but its content will remain intact.

---

8. We shall now extract the image of this container to recreate another container using this new image.

Note: The container may be either running or in stopped state, while we extract the image.

```
osgdev@TG-DevOps-OS004:~$ docker container commit 301a4c3c816c
testimage
sha256:50832d1acf4ca662ab438dc5ac43c5d8704a1e8b7ecdfe4398c8c722dde88794

osgdev@TG-DevOps-OS004:~$ docker image ls
REPOSITORY                                TAG                 IMAGE ID
CREATED             SIZE
testimage                                 latest
50832d1acf4c        27 seconds ago      111MB
tomcat                                    9
4f88b8ccbcd0        2 months ago        714MB
tomcat                                    7
d10641f583b3        2 months ago        456MB
ubuntu                                    16.04
2a4cca5ac898        2 months ago        111MB
hello-world                               latest
f2a91732366c        4 months ago        1.85kB
```

---

9. Create another container using the new "testimage", and check whether it is replica of the "naughty_brown" container, which we modified to add the folder "Master" and file "testfile"

Note: The command /bin/bash is to run bash shell inside the container.

```
osgdev@TG-DevOps-OS004:~$ docker container run -it testimage /bin/bash

root@a092c4f45d64:/# ls
MASTER  boot  etc   lib    media  opt   root  sbin  sys       tmp  var
bin     dev   home  lib64  mnt    proc  run   srv   testfile  usr

osgdev@TG-DevOps-OS004:~/dockerlab$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS                          PORTS
NAMES
a092c4f45d64        testimage           "/bin/bash"              3
minutes ago       Up 3 minutes
keen_bhabha
301a4c3c816c        ubuntu:16.04        "/bin/bash"              41
minutes ago      Exited (0) 21 minutes ago
naughty_brown
```

Note: The newly created container "keen_bhabha" and "naughty_brown" are two different container using different image. But "keen_bhabha" as a container is replica of "naughty_brown" container, as we used the image extracted from that.

10. Container names were generated randomly and hence are irrelevant most of the times. We can rename these containers with more appropriate names.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker container rename
naughty_brown original

osgdev@TG-DevOps-OS004:~/dockerlab$ docker container rename keen_bhabha
replica

osgdev@TG-DevOps-OS004:~/dockerlab$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS                          PORTS
NAMES
a092c4f45d64        testimage           "/bin/bash"             8
minutes ago       Up 8 minutes
replica
301a4c3c816c        ubuntu:16.04        "/bin/bash"             About
an hour ago   Exited (0) 26 minutes ago
original
```

11. Now we have created new image and created another container out of that. How can we share this image with others in the team. One way is we can archive the image in to a single tar file and can share this to team.

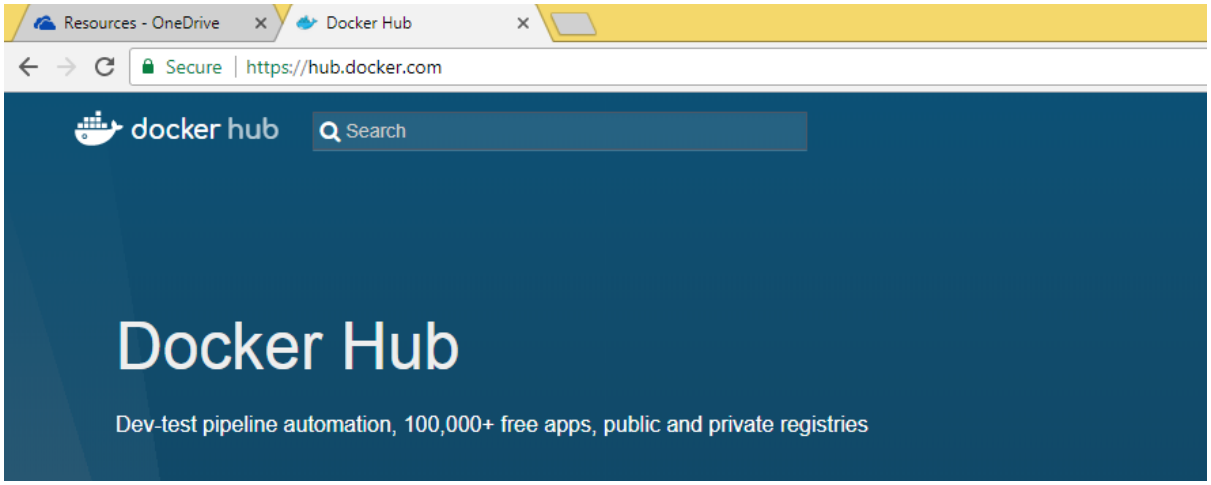
```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker container export -o test.tar
original

osgdev@TG-DevOps-OS004:~/dockerlab$ ls
test.tar
```

You can get image of "original" container and can create another replica.

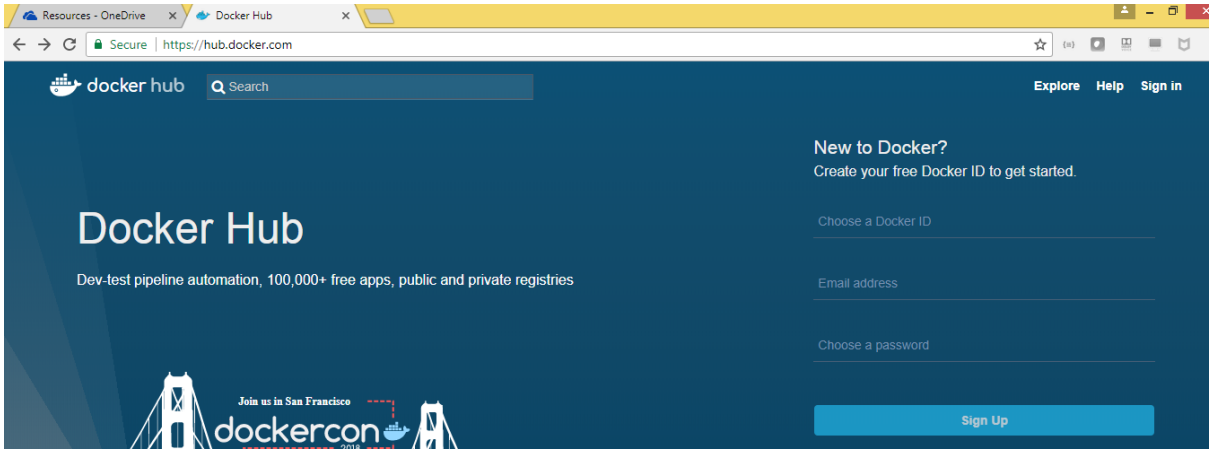
```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker image import ./test.tar
newimage
sha256:b03fc914fdcb021573bdc5c46fc538263d0365e130b398608345f664d97bca44
osgdev@TG-DevOps-OS004:~/dockerlab$ docker image ls
REPOSITORY                              TAG                 IMAGE ID
CREATED             SIZE
newimage                                latest
b03fc914fdcb        7 seconds ago       85.8MB
testimage                               latest
50832d1acf4c        20 minutes ago      111MB

osgdev@TG-DevOps-OS004:~/dockerlab$ docker container run -it --name
replica2 newimage /bin/bash
root@d893a4d0e415:/# ls
MASTER  boot  etc   lib    media  opt   root  sbin  sys       tmp  var
bin     dev   home  lib64  mnt    proc  run   srv   testfile  usr
root@d893a4d0e415:/#
```

Note: Observe that we named the container this time at the time of creation itself.

12. Alternately we can make use of docker hub and upload the image and share the same with anyone.



Create an account for yourself choosing a unique ID, email address (for confirmation) and password (Creating public repo in cloud – No credit card required!). Click on Sign Up. Click on Sign in to access your account.

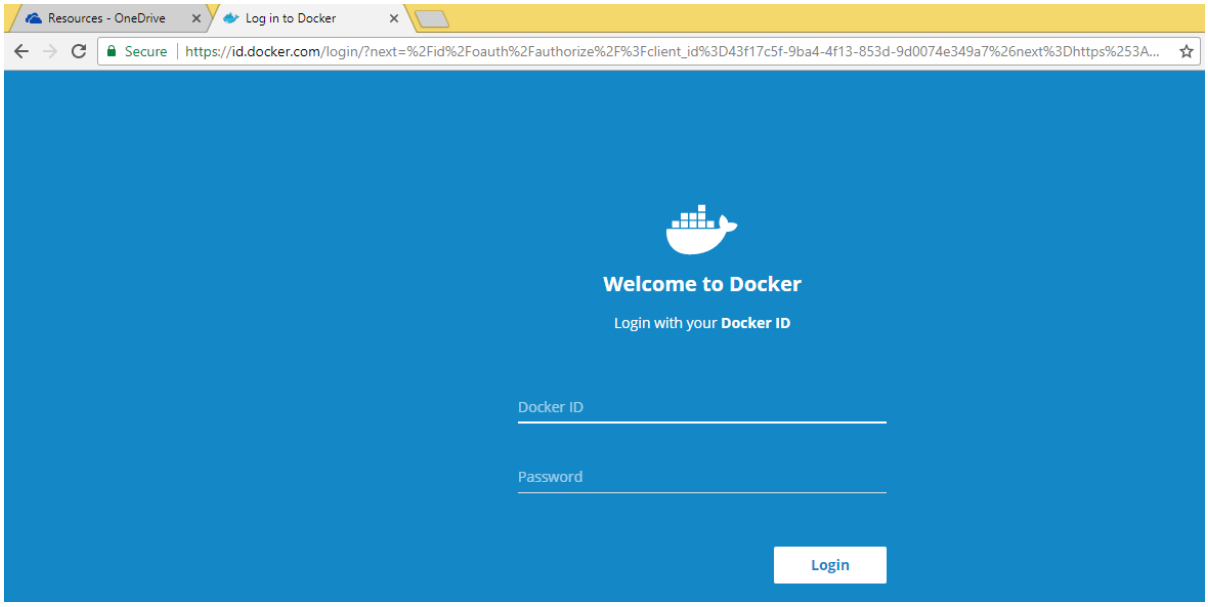



```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If
you don't have a Docker ID, head over to https://hub.docker.com to
create one.
Username: prakaram
Password:
Login Succeeded

osgdev@TG-DevOps-OS004:~/dockerlab$ docker image tag testimage
prakaram/ti001

osgdev@TG-DevOps-OS004:~/dockerlab$ docker image ls
REPOSITORY                                  TAG                 IMAGE ID
CREATED             SIZE
newimage                                    latest
b03fc914fdcb        7 minutes ago       85.8MB
```

```
testimage                                    latest
50832d1acf4c        27 minutes ago      111MB
prakaram/ti001                               latest
50832d1acf4c        27 minutes ago      111MB

osgdev@TG-DevOps-OS004:~/dockerlab$ docker image push prakaram/ti001
The push refers to repository [docker.io/prakaram/ti001]
e8d14772d554: Pushed
8600ee70176b: Mounted from library/ubuntu
2bbb3cec611d: Mounted from library/ubuntu
d2bb1fc88136: Mounted from library/ubuntu
a6a01ad8b53f: Mounted from library/ubuntu
833649a3e04c: Pushed
Head https://registry-
1.docker.io/v2/prakaram/ti001/blobs/sha256:50832d1acf4ca662ab438dc5ac43
c5d8704a1e8b7ecdfe4398c8c722dde88794: read tcp 10.199.0.104:53484-
>35.169.231.249:443: read: connection reset by peer

osgdev@TG-DevOps-OS004:~/dockerlab$ docker logout
Removing login credentials for https://index.docker.io/v1/
```

Note: Here it failed and the image is not uploaded. It depends on the speed of internet connected to your machine.

Sign into your account to view the uploaded image:



---

13. Now anybody can pull this image, as the repository in which I had uploaded the image is a public repository.

Important Note: Make sure that whatever image you upload in this external repository has no important code relating to business, as you may need to explain what you pushed into this external public repository.

For a change I will pull some other image from docker official repository.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker image pull docker/whalesay
Using default tag: latest
latest: Pulling from docker/whalesay
e190868d63f8: Pull complete
909cd34c6fd7: Pull complete
0b9bfabab7c1: Pull complete
a3ed95caeb02: Pull complete
00bf65475aba: Pull complete
c57b6bcc83e3: Pull complete
8978f6879e2f: Pull complete
8eed3712d2cf: Pull complete
Digest:
sha256:178598e51a26abbc958b8a2e48825c90bc22e641de3d31e18aaf55f3258ba93b
Status: Downloaded newer image for docker/whalesay:latest

osgdev@TG-DevOps-OS004:~/dockerlab$ docker image ls
REPOSITORY                                  TAG                 IMAGE ID
CREATED             SIZE
newimage                                    latest
b03fc914fdcb        15 minutes ago      85.8MB
testimage                                   latest
50832d1acf4c        35 minutes ago      111MB
prakaram/ti001                              latest
50832d1acf4c        35 minutes ago      111MB
tomcat                                      9
4f88b8ccbcd0        2 months ago        714MB
tomcat                                      7
d10641f583b3        2 months ago        456MB
ubuntu                                      16.04
2a4cca5ac898        2 months ago        111MB
hello-world                                 latest
f2a91732366c        4 months ago        1.85kB
docker/whalesay                             latest
6b362a9f73eb        2 years ago         247MB

osgdev@TG-DevOps-OS004:~/dockerlab$ docker container run
docker/whalesay cowsay "Always an Important Message"

 _____________________________
< Always an Important Message >
 -----------------------------
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \     \         __/
          \____\_______/
```

---

14. By this time we have too many containers. We shall remove few of them. But you can only remove stopped (Exited) containers.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS                          PORTS
NAMES
a6a9947065c8        docker/whalesay     "cowsay 'Always an I…"   3
minutes ago       Exited (0) 3 minutes ago
hardcore_nightingale
d893a4d0e415        newimage            "/bin/bash"              17
minutes ago      Exited (0) 15 minutes ago
replica2
a092c4f45d64        testimage           "/bin/bash"              34
minutes ago      Up 34 minutes
replica
301a4c3c816c        ubuntu:16.04        "/bin/bash"              About
an hour ago   Exited (0) About an hour ago
original
4f83d502a259        hello-world         "/hello"                 About
an hour ago   Exited (0) About an hour ago
pensive_hugle

osgdev@TG-DevOps-OS004:~/dockerlab$ docker container rm replica2
replica2

osgdev@TG-DevOps-OS004:~/dockerlab$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS                          PORTS
NAMES
a6a9947065c8        docker/whalesay     "cowsay 'Always an I…"   6
minutes ago       Exited (0) 6 minutes ago
hardcore_nightingale
a092c4f45d64        testimage           "/bin/bash"              37
minutes ago      Up 37 minutes
replica
301a4c3c816c        ubuntu:16.04        "/bin/bash"              About
an hour ago   Exited (0) About an hour ago
original
4f83d502a259        hello-world         "/hello"                 About
an hour ago   Exited (0) About an hour ago
pensive_hugle
```

osgdev@TG-DevOps-OS004:~/dockerlab$ docker container rm replica
Error response from daemon: You cannot remove a running container a092c4f45d6468cedf5062782206ac00fefa39f39388ba2aa9141c8e752ecfc7. Stop the container before attempting removal or force remove

Note: If you want to remove multiple containers which are stopped (Exited)

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
a6a9947065c82f4b031b0696285d127f87edac95f1a1c25599d223bf5c7e0b27
301a4c3c816ca4076d4836e52f544bafe891df2cca01ebeaaedf1f81b452f442
4f83d502a2598b6bb90fbb26d106a895c8b1276fa91c60c330de50a954463a6d
fb5b10542ea49648895de7ad3f712e98d847dc35e8d502b189ee20c892be09ca
3bcd41fa2a64acb60e2e81d735d68ba31b812ffee791c3a90bf88eb88eeb945f
91cf99b10039ddd32dab7c047ef41c494ca568629bc5cf763d37b9775cb058ae
```

```
Total reclaimed space: 47B
osgdev@TG-DevOps-OS004:~/dockerlab$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
a092c4f45d64        testimage           "/bin/bash"         38 minutes
ago      Up 38 minutes                           replica

Note: To remove the container "replica", you first need to stop it.

osgdev@TG-DevOps-OS004:~/dockerlab$ docker container stop replica
replica

osgdev@TG-DevOps-OS004:~/dockerlab$ docker container rm replica
replica

osgdev@TG-DevOps-OS004:~/dockerlab$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
```

---

15. We can also remove images in similar way. But if there is any image used in a container then it cannot be removed.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker container run testimage

osgdev@TG-DevOps-OS004:~/dockerlab$ docker image ls
REPOSITORY                                  TAG                 IMAGE ID
CREATED             SIZE
newimage                                    latest
b03fc914fdcb        2 hours ago         85.8MB
testimage                                   latest
50832d1acf4c        3 hours ago         111MB
prakaram/ti001                              latest
50832d1acf4c        3 hours ago         111MB

osgdev@TG-DevOps-OS004:~/dockerlab$ docker image rm newimage
Untagged: newimage:latest
Deleted:
sha256:b03fc914fdcb021573bdc5c46fc538263d0365e130b398608345f664d97bca44
Deleted:
sha256:133188bd6d7b80d00f96de214b5d30a1b73aa39a02d7dcdc6b5d9afcd0619072

osgdev@TG-DevOps-OS004:~/dockerlab$ docker image ls
REPOSITORY                                  TAG                 IMAGE ID
CREATED             SIZE
prakaram/ti001                              latest
50832d1acf4c        3 hours ago         111MB

osgdev@TG-DevOps-OS004:~/dockerlab$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS                      PORTS               NAMES
7b352091bb4c        50832d1acf4c        "/bin/bash"         3 minutes
ago       Exited (0) 3 minutes ago
wizardly_kepler
```

```
06f4dfd5ddfb        ubuntu:16.04        "/bin/bash"         4 minutes
ago       Exited (0) 4 minutes ago
practical_shannon

osgdev@TG-DevOps-OS004:~/dockerlab$ docker image rm prakaram/ti001
Error response from daemon: conflict: unable to remove repository
reference "prakaram/ti001" (must force) - container 7b352091bb4c is
using its referenced image 50832d1acf4c

sgdev@TG-DevOps-OS004:~/dockerlab$ docker container rm 7b352091bb4c
7b352091bb4c
osgdev@TG-DevOps-OS004:~/dockerlab$ docker image rm prakaram/ti001
Untagged: prakaram/ti001:latest
Deleted:
sha256:50832d1acf4ca662ab438dc5ac43c5d8704a1e8b7ecdfe4398c8c722dde88794
Deleted:
sha256:bc773095ee15bfa8022615c51f2d43f0b9f78dd7de650b251be6dc86f7b6f38e
```

To remove multiple unused images.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker image prune -a
WARNING! This will remove all images without at least one container
associated to them.
Are you sure you want to continue? [y/N] y
Deleted Images:
untagged: jenkinsnew01:lts
deleted: sha256:63a943d4cf285e6ced91f655f08e69ae9cfc00b5cb8aee101f34a11bc50903c1
untagged: jenkinsnew:lts
untagged: prakaram/jenkins:latest
untagged:
prakaram/jenkins@sha256:3e9a6518c0860aa8f340fa85a039444e81e4a749f148e53988dbd8e5b6e
1a1ae
deleted: sha256:0147c2cdd418eaac9a442aad91e3f69d767e7402454e99e6c8e6cc83b3749612
deleted: sha256:3bb82ee42670c342f55bccf504d6d5d4e259f6bbb426f9c6edadccec13e9aef1
untagged: hello-world:latest
untagged: hello-
world@sha256:66ef312bbac49c39a89aa9bcc3cb4f3c9e7de3788c944158df3ee0176d32b751
deleted: sha256:f2a91732366c0332ccd7afd2a5c4ff2b9af81f549370f7a19acd460f87686bc7
deleted: sha256:f999ae22f308fea973e5a25b57699b5daf6b0f1150ac2a5c2ea9d7fecee50fdf
untagged: tomcat:9
untagged:
tomcat@sha256:a8fccd0c77044339379e7a6faf1bbebc93b84a3f4d6c6ecf0834ed9c11eaafaf
deleted: sha256:4f88b8ccbcd0109e924a01a649f18426cdaee907b6b90f2d7c494a001dff32b4
deleted: sha256:d7e2fe0506e99ef05fa0e40029992b167615c8bbe4489fbe7b9c17d78ef0045a
deleted: sha256:1fe811e1fabb4e2a22e687849a0e0222e1a237ece29b64e33e422e587f242eb4
deleted: sha256:c77725c8ae4d9f495727cdf6cf1307f0c2cea822f74f32fc7073b263ee25eaa9
deleted: sha256:2d3c80ba7421a374d81ac7e2ebf3a354f1174539f29b0e44c75957cdcc6b2515
deleted: sha256:86049a20c74215a3e97b755f9be2a97ced4d640f21f0f174486c507184ccd199
deleted: sha256:2adfce4fa591bca57af58674dd94c25abd6dad67f244834ad0eb24ffa77eec21
deleted: sha256:263b8cdbb9538f5fb2259767f49e375b2f85efdda0bf43a13856e882fdfce6fa
deleted: sha256:9bb7205feda00603455c9911a744b48842d390e0588f92da82ec802a38fb51c7
deleted: sha256:af1087a0c804e99c4f0a45754d2175d854d9688ffce6da36cf1f96798e7f50c3
deleted: sha256:c67caef9098e426d75b7de6dbad44fac70d445bb6bb9c8bcb276392efe279853
deleted: sha256:d4357ae863b2e679ed163cef9d5361719666a39d56cce124993f3eed52f5e5ae
untagged: docker.bintray.io/jfrog/artifactory-oss:latest
untagged: docker.bintray.io/jfrog/artifactory-
oss@sha256:7cbda8f30cea41556a52ea27239c9425ba9d305f6af0a0cc1d675279855e7477
deleted: sha256:8456545abc598ead6e3113f683bdcda40f1a85a964356a9aee54407c531ea82b
deleted: sha256:a378220bbe6a1edeb7f179f6bd9b86c9c156bb870a37e403657cabd2f8b09593
deleted: sha256:87e284c79d68b2ed1c05c2f8545117e3ee08c196efa71290862dfca8725cc518
deleted: sha256:1a5fd27f38b9e4ffdc365441a749ed0c6d2427ba49cf958d0b16be4eb2112020
deleted: sha256:70bb1529a6727b7744c7cb2e3f0f4be9b267ef2311d6f5a623225ec402071b13
```

```
deleted: sha256:bae6e95df1dc7e4c3d14103229928defb90ee3d62f0ccabc68e435672b44043d
deleted: sha256:c7a2e0d0dac7725c6d7f51bd0455e1598a41626325245c4b3eff3a5f6c0699fb
deleted: sha256:6e8f24a7751eabebed9b97c6a8fe1fec799525cafee36d3f2970ecbd31a3091f
deleted: sha256:514a17b94edbfa0eaade7cb5292ea7ad361a49171cae8d636d1e0ef5ab6a5ea7
deleted: sha256:d7ebb1ef5f772b588c6050af418991994ca66ab67010d1cf57fa6ed4ede3ac52
deleted: sha256:221fc56c08ea085cf07093bbec010c88be38ca8a3d6485e69259cacf3a7d477b
deleted: sha256:78aef494d378ebe8924ca453b39109b104df08ac30d511004c09391c5f26e013
deleted: sha256:bd7109b3ddc2e32341649711c41d69b7a9cbed248ffa2ed7c3258c2184a7b8e1
deleted: sha256:9590247881214d4e9516637e56fcf673dcdd81ba469ba99def5606eb926d5dc3
deleted: sha256:356a2093ce3fdc09fff7cb4393ab625a16d2c773f0a0616c25bac32a76f924bf
deleted: sha256:eaa640d739c173a294c1d77f4db89812d136ec20ed2592ae38a04bd01c3d0d4a
deleted: sha256:1e1a7f351c46ae71c98446b4decb9b67d08ea624e62324760f565b436b9bf305
deleted: sha256:1599365ecbaf5b106d874fcbce0a70283e3ecff771f0f826c7695fd85c2f7eaa
deleted: sha256:e46edc61693239a668ff0c341119e9563e66bb6cd9a6d09b89b0eaec26698723
deleted: sha256:cf4ecb49238476635f551fe11987ae4c374a22a475ffc3b69215b3fe748f5235
untagged: tomcat:7
untagged:
tomcat@sha256:7ab44b588ca355f26b60fc0212a0f5a43935ee4375df9105a02a84ac6b3e59af
deleted: sha256:d10641f583b3e74da83d86cb97d1a1eb45475cafe4da4d2c3ace22354c1b4b4a
deleted: sha256:7311bf8908a0617e0bbd8360a12e0df91b80ed459b0144871d1a3d4260fd07f2
deleted: sha256:4cfcdcd79e6061e204dacbfe28dd2063f7d1deae7000ee0da14dc5f1ac369248
deleted: sha256:72bc7ec090723622fe946e1b51c15b502165a949afa875f0a9d0c17eb9dcdaed
deleted: sha256:b928545d34b66db92c5b0b45b0dc23325a2aa9ec6d8c872ed6e8da7a7c7845c4
deleted: sha256:32ceabae6c60346c42a505358cf8ade36163bd5a4ab25c31df8e3f62b7f0b99b
deleted: sha256:da08855baba5e0eecbf81844082ce0ce55daadf168e52b33f6d3697f949c5590
deleted: sha256:c37b4bb3a823add102bda0bd609d00a01dc330046691a82e2c8d12b4a62bedb8
deleted: sha256:39bbb91a1134b6e7f7a382eb9451d4a993de0bf0cf43065b30ee1323e93c0f56
deleted: sha256:c9b9ac13253e900e46c500b2153a88c571cb00fce1b0a086ce43482b10eace4b
deleted: sha256:7152b3dd92f68f075f10409f8a0060cc1e2cfe05bc39e9066d6962ab4fe16ccc
deleted: sha256:4bcdffd70da292293d059d2435c7056711fab2655f8b74f48ad0abe042b63687
untagged: docker/whalesay:latest
untagged:
docker/whalesay@sha256:178598e51a26abbc958b8a2e48825c90bc22e641de3d31e18aaf55f3258b
a93b
deleted: sha256:6b362a9f73eb8c33b48c95f4fcce1b6637fc25646728cf7fb0679b2da273c3f4
deleted: sha256:34dd66b3cb4467517d0c5c7dbe320b84539fbb58bc21702d2f749a5c932b3a38
deleted: sha256:52f57e48814ed1bb08a651ef7f91f191db3680212a96b7f318bff0904fed2e65
deleted: sha256:72915b616c0db6345e52a2c536de38e29208d945889eecef01d0fef0ed207ce8
deleted: sha256:4ee0c1e90444c9b56880381aff6455f149c92c9a29c3774919632ded4f728d6b
deleted: sha256:86ac1c0970bf5ea1bf482edb0ba83dbc88fefb1ac431d3020f134691d749d9a6
deleted: sha256:5c4ac45a28f91f851b66af332a452cba25bd74a811f7e3884ed8723570ad6bc8
deleted: sha256:088f9eb16f16713e449903f7edb4016084de8234d73a45b1882cf29b1f753a5a
deleted: sha256:799115b9fdd1511e8af8a8a3c8b450d81aa842bbf3c9f88e9126d264b232c598
deleted: sha256:3549adbf614379d5c33ef0c5c6486a0d3f577ba3341f573be91b4ba1d8c60ce4
deleted: sha256:1154ba695078d29ea6c4e1adb55c463959cd77509adf09710e2315827d66271a
untagged: jenkins:lts
untagged: jenkins/jenkins:lts
untagged:
jenkins/jenkins@sha256:3fc5ccef3d5851b61a5e9b850c338b03ec1b61c8a830d280b111fef229d1
e847
deleted: sha256:279f21046a63d2246e62d7b17c620fb9dc5103d579aff9c3813476b36f4b62d5
deleted: sha256:846f5260d574472fc4be3b5bc7a6c3cb76edc66720da2fd15deefb87f639bd7c
deleted: sha256:b68b2b56060b174fc0b604a70540934149000fb5853791cd526caf88cdc4ddad
deleted: sha256:383a5cf8b8cded1e425f4ccf2ab7558eb2648dff825b83a11cf850d682da1f43
deleted: sha256:31c393c3846a3e4dacbabf0ce04d8a9a8fec6830955944f406f89827a4168734
deleted: sha256:5f496144a525f443ab9ad4665bf8759e6fe315bd8f38f05a328c0ecde930c602
deleted: sha256:f22e610372acaa978c5df5613a8de2a13d7691ccba1d5fe7d52ea3f7045382ec
deleted: sha256:d39948cf2c43ec399547eee448b123f3e39fcbe85660841dd690b748089afa36
deleted: sha256:e88f37b6e7bfc1a0a565d53161c7e2b52e290bea147c243df5e42a0176d5b4d8
deleted: sha256:0237b0ca72c991be736b4a850aa9e02389e99185d17faafd0e9eb261514254bc
deleted: sha256:5e9c3dd7307d42ead1dd8e0463686f8624680017f9414db55b83940dfdc54164
deleted: sha256:1a9280c141b892c5cad1f55e94a41aa862975a965e92a073659d9cdc8da53f9d
deleted: sha256:18bddc758e3e018d3809a04d283af0058eb252ca020ecfd4bc1558f14ea80fab
deleted: sha256:3b1665aab3a2390fe12d521809c04381319895479858ee3d5a73ebd1f73f9174
deleted: sha256:24f7afc0231f18aafee496b1c4787c6f50576581b61a203db161a3c27b1e1f90
deleted: sha256:fc979870700392254f8ee3e58699535f18d647f1607a5557238d24b4de2b698a
deleted: sha256:85d35c8fad42698ad3688ab07046144f49d5a8ccc866260fc75309d4b2ad724f
deleted: sha256:35b743766cf969889cbb3467c7cfa36bdfc604e4e945f446cc96c8cf589a5ce1
```

```
deleted: sha256:1e427582a181f94999e09606d2b6dac986b3f2c20be1ebc351f22afe20d01df6
deleted: sha256:a6d2a4d6116a954ef3a8b0db04d785571b7b2c2e3f8e7990e862b11a0deebb1b
deleted: sha256:e27a10675c5656bafb7bfa9e4631e871499af0a5ddfda3cebc0ac401dfe19382
untagged: jenkinsnew02:lts
deleted: sha256:66be4f0b83b34f46be1e821979ae52c0fc0d0f6c4db1db816abaa2542aa9cbc6
deleted: sha256:2a00a0903b6bf7d0230b2ebee3fa78a75836137cd8a5bba297e4c314f3643e14
deleted: sha256:568e40e2bb51b00cd9769ba0350e55fc572d9631bb3eeb839c8762506c54cc04
deleted: sha256:665aabd8c5160398aec712b3f13fb64b129cae1853ac9613ec1c67e02aef600e
deleted: sha256:738481bcfaecc62d1dd5da7f70e9f0af8ec7c3e814330bd5bae98ca96a786d78

Total reclaimed space: 4.147GB
osgdev@TG-DevOps-OS004:~/dockerlab$

osgdev@TG-DevOps-OS004:~/dockerlab$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
ubuntu              16.04               2a4cca5ac898        2 months
ago         111MB
osgdev@TG-DevOps-OS004:~/dockerlab$
```

---

16. Docker general subcommands. Track how many of them you know till now. Note that this is the old way of docker command line which makes no distinction between container or image specific commands.

Note: This training manual do not use this old style docker COMMAND. But similar activity is covered under the new docker container COMMAND and docker image COMMAND listed in the following two steps.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker

Usage:      docker COMMAND
```

A self-sufficient runtime for containers

```
Options:
      --config string      Location of client config files (default
                           "/home/osgdev/.docker")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level
                           ("debug"|"info"|"warn"|"error"|"fatal")
                           (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default
                           "/home/osgdev/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default
                           "/home/osgdev/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default
                           "/home/osgdev/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Management Commands:
  config      Manage Docker configs
  container   Manage containers
  image       Manage images
  network     Manage networks
```

```
  node        Manage Swarm nodes
  plugin      Manage plugins
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  trust       Manage trust on Docker images (experimental)
  volume      Manage volumes

Commands:
  attach      Attach local standard input, output, and error streams to
a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local
filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's
filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem
image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the
container
  ps          List containers
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  run         Run a command in a new container
  save        Save one or more images to a tar archive (streamed to
STDOUT by default)
  search      Search the Docker Hub for images
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage
statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  version     Show the Docker version information
```

```
  wait        Block until one or more containers stop, then print their
exit codes

Run 'docker COMMAND --help' for more information on a command.
```

---

17. Docker Container specific subcommands. Track how many of them you know till now.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker container

Usage:     docker container COMMAND

Manage containers

Options:


Commands:
  attach      Attach local standard input, output, and error streams to
a running container
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local
filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's
filesystem
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  inspect     Display detailed information on one or more containers
  kill        Kill one or more running containers
  logs        Fetch the logs of a container
  ls          List containers
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the
container
  prune       Remove all stopped containers
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  run         Run a command in a new container
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage
statistics
  stop        Stop one or more running containers
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  wait        Block until one or more containers stop, then print their
exit codes

Run 'docker container COMMAND --help' for more information on a
command.
```

---

18. Docker Image specific subcommands. Track how many of them you know till now.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker image

Usage:     docker image COMMAND

Manage images

Options:


Commands:
  build       Build an image from a Dockerfile
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem
image
  inspect     Display detailed information on one or more images
  load        Load an image from a tar archive or STDIN
  ls          List images
  prune       Remove unused images
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rm          Remove one or more images
  save        Save one or more images to a tar archive (streamed to
STDOUT by default)
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Run 'docker image COMMAND --help' for more information on a command.
```

19.

20.

21.
