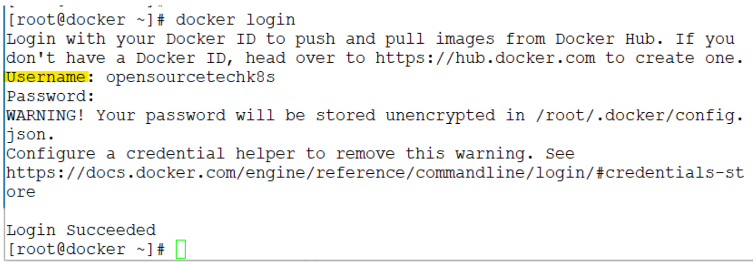
02 January 2021 19:29

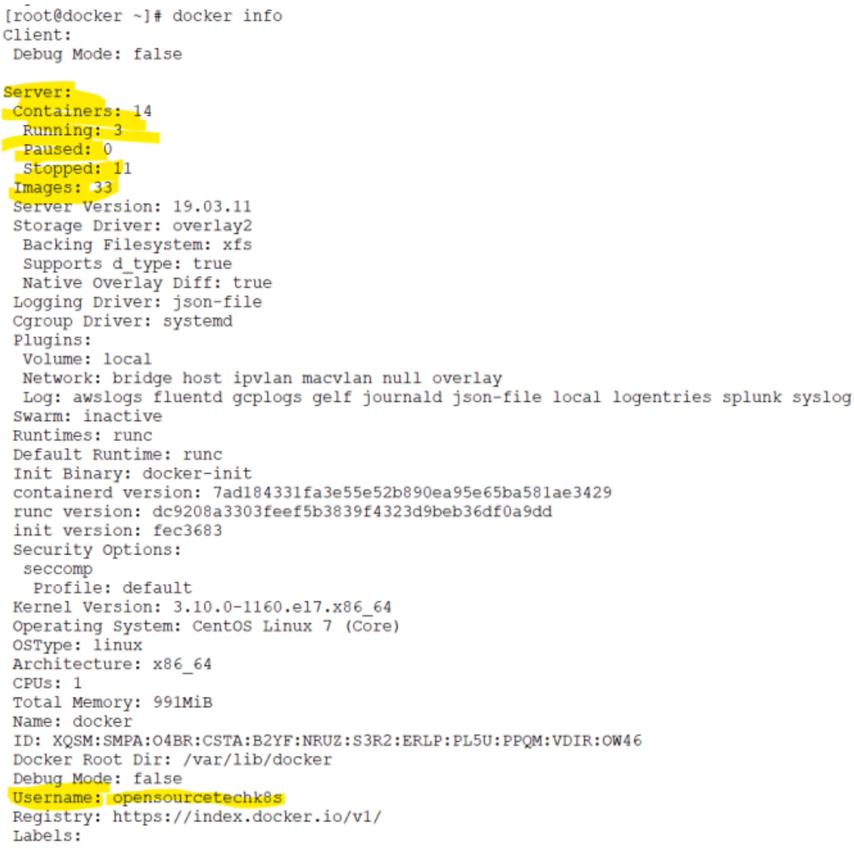
LAB

- **Create a id on docker HUB**
- **Login with that id on your local docker instance**



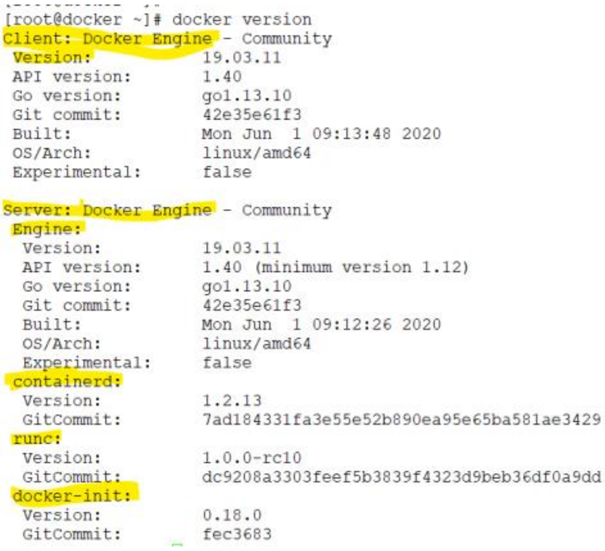
```
[root@docker ~]# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you
don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: opensourcetechk8s
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.
json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-st
ore

Login Succeeded
[root@docker ~]#
```

- **Verify you login and state of docker resources (container, Images, storage)**



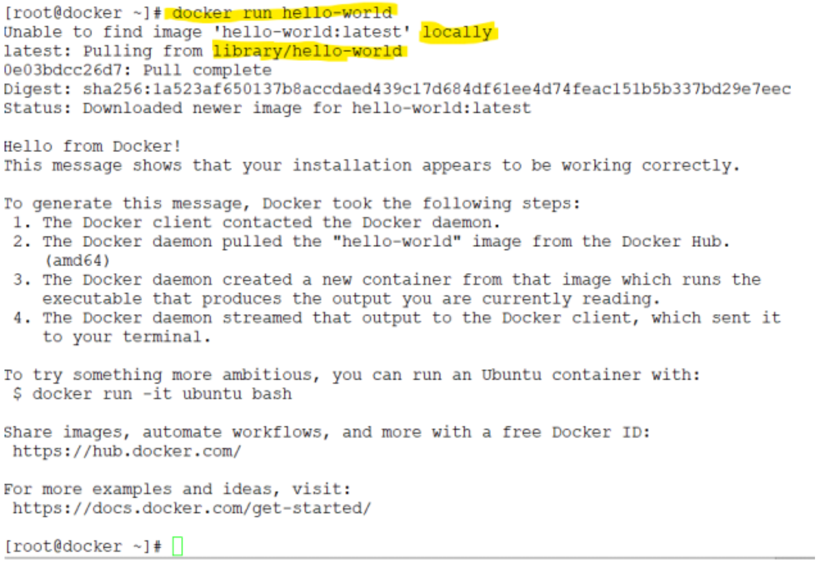
```
[root@docker ~]# docker info
Client:
 Debug Mode: false

Server:
 Containers: 14
  Running: 3
  Paused: 0
  Stopped: 11
 Images: 33
 Server Version: 19.03.11
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: systemd
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc version: dc9208a3303feef5b3839f4323d9beb36df0a9dd
 init version: fec3683
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 3.10.0-1160.el7.x86_64
 Operating System: CentOS Linux 7 (Core)
 OSType: linux
 Architecture: x86_64
 CPUs: 1
 Total Memory: 991MiB
 Name: docker
 ID: XQSM:SMPA:O4BR:CSTA:B2YF:NRUZ:S3R2:ERLP:PL5U:PPQM:VDIR:OW46
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Username: opensourcetechk8s
 Registry: https://index.docker.io/v1/
 Labels:
```

- **Check docker version and list of components used by Docker**
    - **These are the key features docker provides in comparison with LXC**



```
[root@docker ~]# docker version
Client: Docker Engine - Community
 Version:           19.03.11
 API version:       1.40
 Go version:        go1.13.10
 Git commit:        42e35e61f3
 Built:             Mon Jun  1 09:13:48 2020
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.11
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.13.10
  Git commit:       42e35e61f3
  Built:            Mon Jun  1 09:12:26 2020
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.2.13
  GitCommit:        7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc:
  Version:          1.0.0-rc10
  GitCommit:        dc9208a3303feef5b3839f4323d9beb36df0a9dd
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
```

- **Run a command to test your docker installation and check the output:**
    - **Always docker tries to get image locally, if it doesn't exist than docker will try to pull from docker HUB**

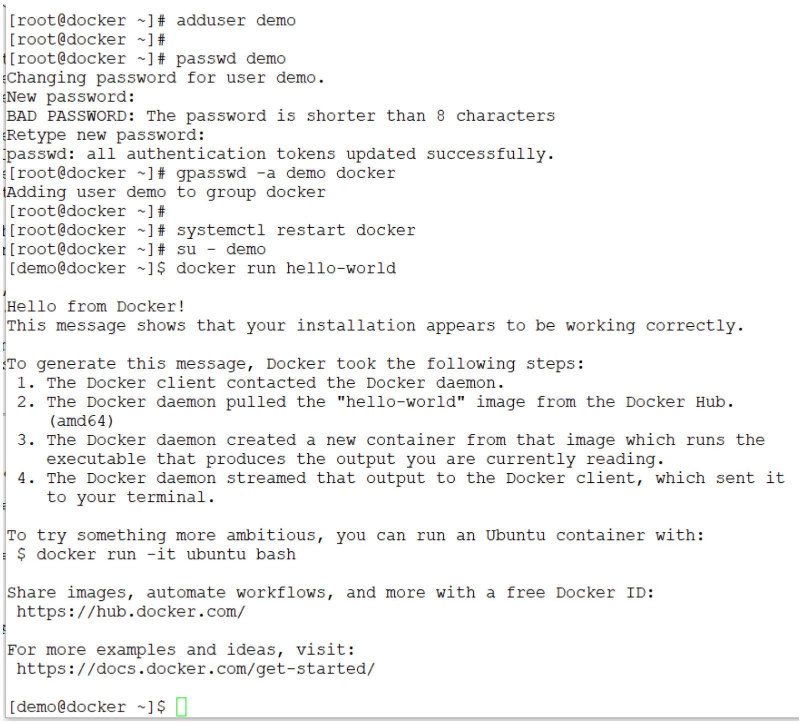


```
[root@docker ~]# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:1a523af650137b8accdaed439c17d684df61ee4d74feac151b5b337bd29e7eec
Status: Downloaded newer image for hello-world:latest

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://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

[root@docker ~]# 
```

NOTE: **The docker daemon binds to a Unix socket instead of a TCP port. By default that Unix socket is owned by the user root and other users can only access it using sudo. The docker daemon always runs as the root user.**

If you don't want to use sudo when you use the docker command, create a Unix group called docker and add users to it. When the docker daemon starts, it makes the ownership of the Unix socket read/writable by the docker group.

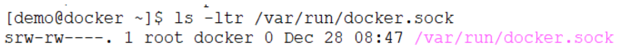
Warning: The docker group grants privileges equivalent to the root user. For details on how this impacts security in your system, see *Docker Daemon Attack Surface*.



```
[root@docker ~]# adduser demo
[root@docker ~]#
[root@docker ~]# passwd demo
Changing password for user demo.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
[root@docker ~]# gpasswd -a demo docker
Adding user demo to group docker
[root@docker ~]#
[root@docker ~]# systemctl restart docker
[root@docker ~]# su - demo
[demo@docker ~]$ 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://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

[demo@docker ~]$ 
```

- Check the ownership of the docker socket file

```
# ls –ltr /var/run/docker.sock
```



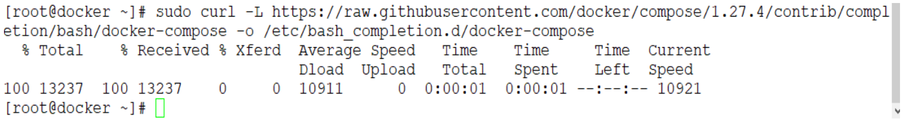
```
[demo@docker ~]$ ls -ltr /var/run/docker.sock
srw-rw----. 1 root docker 0 Dec 28 08:47 /var/run/docker.sock
```

- **Docker command auto completion:**

LINUX

1) On a current Linux OS (in a non-minimal installation), bash completion should be available.
2) Place the completion script in /etc/bash_completion.d/.

```
# yum -y install bash-completion
# sudo curl -L https://raw.githubusercontent.com/docker/compose/1.27.4/contrib/completion/bash/docker-compose -o /etc/bash_completion.d/docker-compose
```

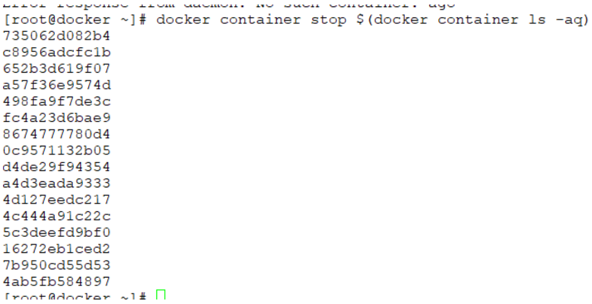


```
[root@docker ~]# sudo curl -L https://raw.githubusercontent.com/docker/compose/1.27.4/contrib/compl
etion/bash/docker-compose -o /etc/bash_completion.d/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 13237  100 13237    0     0  10911      0  0:00:01  0:00:01 --:--:-- 10921
[root@docker ~]# 
```

3) Logout and login again.

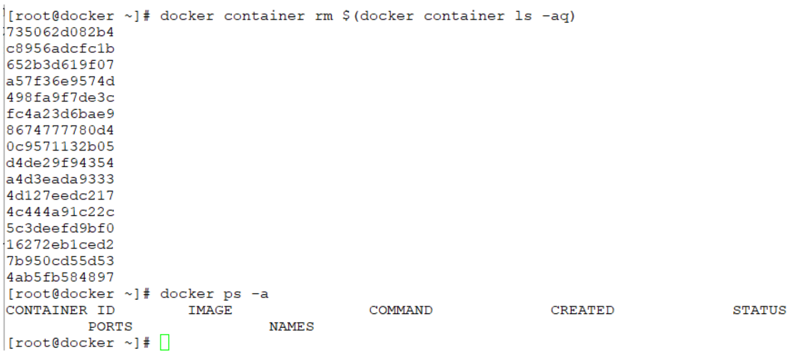
- **Run a Container**
- **Connect to the service running inside your container**

- **List all stopped containers**



```
Error response from daemon: No such container: ago
[root@docker ~]# docker container stop $(docker container ls -aq)
735062d082b4
c8956adcfc1b
652b3d619f07
a57f36e9574d
498fa9f7de3c
fc4a23d6bae9
8674777780d4
0c9571132b05
d4de29f94354
a4d3eada9333
4d127eedc217
4c444a91c22c
5c3deefd9bf0
16272eb1ced2
7b950cd55d53
4ab5fb584897
[root@docker ~]#
```

- **Delete all stopped containers**



```
[root@docker ~]# docker container rm $(docker container ls -aq)
735062d082b4
c8956adcfc1b
652b3d619f07
a57f36e9574d
498fa9f7de3c
fc4a23d6bae9
8674777780d4
0c9571132b05
d4de29f94354
a4d3eada9333
4d127eedc217
4c444a91c22c
5c3deefd9bf0
16272eb1ced2
7b950cd55d53
4ab5fb584897
[root@docker ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
        PORTS               NAMES
[root@docker ~]#
```

- **Create a test container and run following commands**

```
# docker run -d -p 8080:80 --name http-lab centos/httpd
# docker ps -a
```

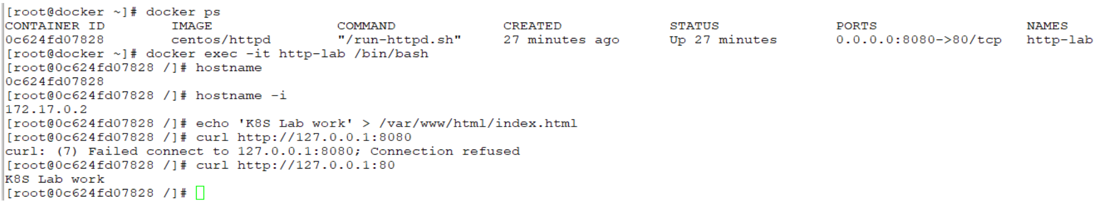
- **Customize the http-lab container to display Hello World as the message**

```
[root@node1 ~]# docker ps
CONTAINER ID   IMAGE        COMMAND          CREATED         STATUS        PORTS                  NAMES
84bc66a19992   centos/httpd   "/run-httpd.sh"   3 minutes ago   Up 3 minutes   0.0.0.0:8080->80/tcp   http-lab
[root@node1 ~]# docker exec -it http-lab /bin/bash
[root@84bc66a19992 /]# vi /var/www/html/index.html
[root@84bc66a19992 /]# ls -l /var/www/html/index.html
-rw-r--r--. 1 root root 13 Dec 26 09:18 /var/www/html/index.html
[root@84bc66a19992 /]# cat /var/www/html/index.html
Hello World!
[root@84bc66a19992 /]# exit
exit
[root@node1 ~]# curl http://localhost:8080
Hello World!
[root@node1 ~]#
```

- **Pull the container ip address and check the http webpage**

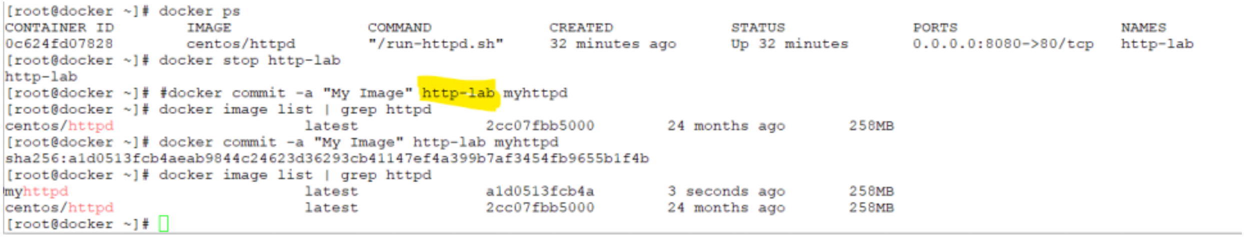
```
[root@node1 ~]# docker inspect -f '{{ .NetworkSettings.IPAddress }}' http-lab
172.17.0.2
[root@node1 ~]# curl http://172.17.0.2
Hello World!
[root@node1 ~]#
```

- **Custom images**
    - **Create image from running container**



```
[root@docker ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS                  NAMES
0c624fd07828        centos/httpd        "/run-httpd.sh"     27 minutes ago      Up 27 minutes       0.0.0.0:8080->80/tcp   http-lab
[root@docker ~]# docker exec -it http-lab /bin/bash
[root@0c624fd07828 /]# hostname
0c624fd07828
[root@0c624fd07828 /]# hostname -i
172.17.0.2
[root@0c624fd07828 /]# echo 'K8S Lab work' > /var/www/html/index.html
[root@0c624fd07828 /]# curl http://127.0.0.1:8080
curl: (7) Failed connect to 127.0.0.1:8080; Connection refused
[root@0c624fd07828 /]# curl http://127.0.0.1:80
K8S Lab work
[root@0c624fd07828 /]#
```

- **Log out and stop the container (recommended)**



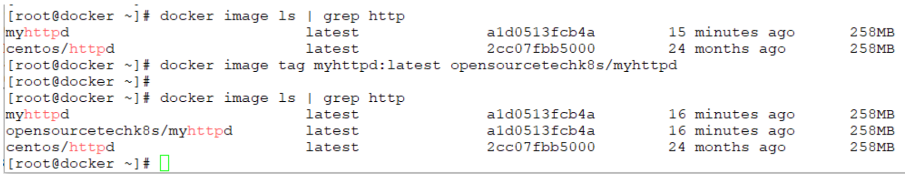
```
[root@docker ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS                  NAMES
0c624fd07828        centos/httpd        "/run-httpd.sh"     32 minutes ago      Up 32 minutes       0.0.0.0:8080->80/tcp   http-lab
[root@docker ~]# docker stop http-lab
http-lab
[root@docker ~]# #docker commit -a "My Image" http-lab myhttpd
[root@docker ~]# docker image list | grep httpd
centos/httpd                   latest              2cc07fbb5000        24 months ago       258MB
[root@docker ~]# docker commit -a "My Image" http-lab myhttpd
sha256:a1d0513fcb4aeab9844c24623d36293cb41147ef4a399b7af3454fb9655b1f4b
[root@docker ~]# docker image list | grep httpd
myhttpd                        latest              a1d0513fcb4a        3 seconds ago       258MB
centos/httpd                   latest              2cc07fbb5000        24 months ago       258MB
[root@docker ~]#
```

- **Push image to docker**
    - **Create a repo on docker hub**

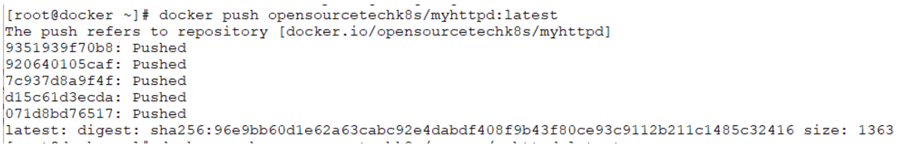
Syntax to push

- **Tag your image with repository path is must**

# docker push <dockerloginid>/<image name>

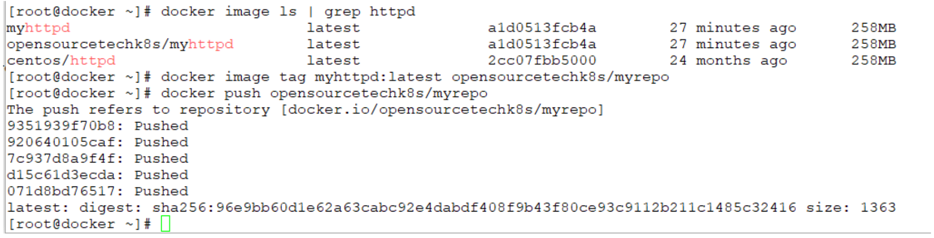


```
[root@docker ~]# docker image ls | grep http
myhttpd                        latest              a1d0513fcb4a        15 minutes ago      258MB
centos/httpd                   latest              2cc07fbb5000        24 months ago       258MB
[root@docker ~]# docker image tag myhttpd:latest opensourcetechk8s/myhttpd
[root@docker ~]#
[root@docker ~]# docker image ls | grep http
myhttpd                        latest              a1d0513fcb4a        16 minutes ago      258MB
opensourcetechk8s/myhttpd      latest              a1d0513fcb4a        16 minutes ago      258MB
centos/httpd                   latest              2cc07fbb5000        24 months ago       258MB
[root@docker ~]#
```

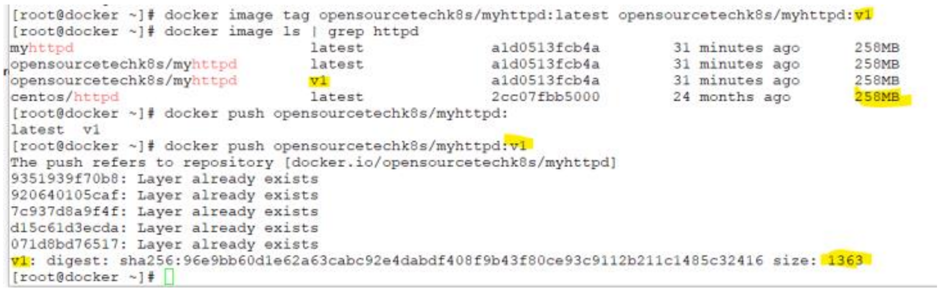


```
[root@docker ~]# docker push opensourcetechk8s/myhttpd:latest
The push refers to repository [docker.io/opensourcetechk8s/myhttpd]
9351939f70b8: Pushed
920640105caf: Pushed
7c937d8a9f4f: Pushed
d15c61d3ecda: Pushed
071d8bd76517: Pushed
latest: digest: sha256:96e9bb60d1e62a63cabc92e4dabdf408f9b43f80ce93c9112b211c1485c32416 size: 1363
```

- **Push image in your repo on docker hub**
  - **Synatx**
    - **Tag your image**

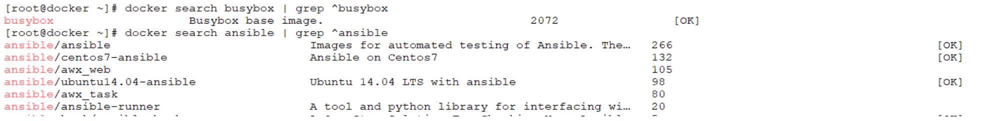


```
[root@docker ~]# docker image ls | grep httpd
myhttpd                             latest              a1d0513fcb4a        27 minutes ago      258MB
opensourcetechk8s/myhttpd           latest              a1d0513fcb4a        27 minutes ago      258MB
centos/httpd                        latest              2cc07fbb5000        24 months ago       258MB
[root@docker ~]# docker image tag myhttpd:latest opensourcetechk8s/myrepo
[root@docker ~]# docker push opensourcetechk8s/myrepo
The push refers to repository [docker.io/opensourcetechk8s/myrepo]
9351939f70b8: Pushed
920640105caf: Pushed
7c937d8a9f4f: Pushed
d15c61d3ecda: Pushed
071d8bd76517: Pushed
latest: digest: sha256:96e9bb60d1e62a63cabc92e4dabdf408f9b43f80ce93c9112b211c1485c32416 size: 1363
[root@docker ~]# 
```



```
[root@docker ~]# docker image tag opensourcetechk8s/myhttpd:latest opensourcetechk8s/myhttpd:v1
[root@docker ~]# docker image ls | grep httpd
myhttpd                             latest              a1d0513fcb4a        31 minutes ago      258MB
opensourcetechk8s/myhttpd           latest              a1d0513fcb4a        31 minutes ago      258MB
opensourcetechk8s/myhttpd           v1                  a1d0513fcb4a        31 minutes ago      258MB
centos/httpd                        latest              2cc07fbb5000        24 months ago       258MB
[root@docker ~]# docker push opensourcetechk8s/myhttpd:
latest  v1
[root@docker ~]# docker push opensourcetechk8s/myhttpd:v1
The push refers to repository [docker.io/opensourcetechk8s/myhttpd]
9351939f70b8: Layer already exists
920640105caf: Layer already exists
7c937d8a9f4f: Layer already exists
d15c61d3ecda: Layer already exists
071d8bd76517: Layer already exists
v1: digest: sha256:96e9bb60d1e62a63cabc92e4dabdf408f9b43f80ce93c9112b211c1485c32416 size: 1363
[root@docker ~]# 
```

-
- **How to search images** - search images on docker hub
  https://docs.docker.com/engine/reference/commandline/search/



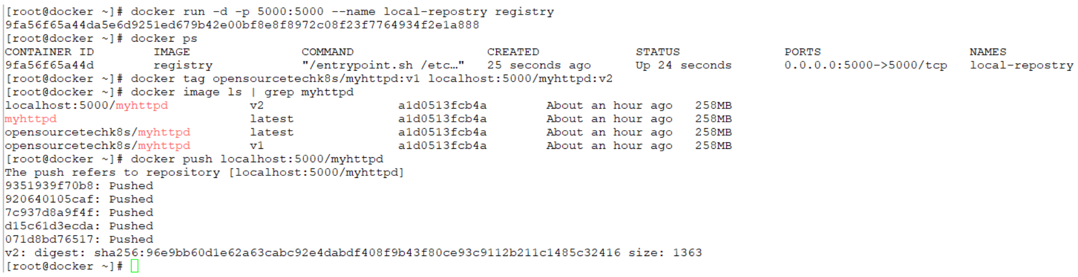
```
[root@docker ~]# docker search busybox | grep ^busybox
busybox                   Busybox base image.                               2072                [OK]
[root@docker ~]# docker search ansible | grep ^ansible
ansible/ansible                       Images for automated testing of Ansible. The…   266                                    [OK]
ansible/centos7-ansible               Ansible on Centos7                              132                                    [OK]
ansible/awx_web                                                                       105
ansible/ubuntu14.04-ansible           Ubuntu 14.04 LTS with ansible                   98                                     [OK]
ansible/awx_task                                                                      80
ansible/ansible-runner                A tool and python library for interfacing wi…   20
```

Local Registry:

The registry image is configured to start on port 5000 in the container, so we will expose the host port also as 5000. Use below command to download registry image from Docker HUB to Docker Host and launch a container.

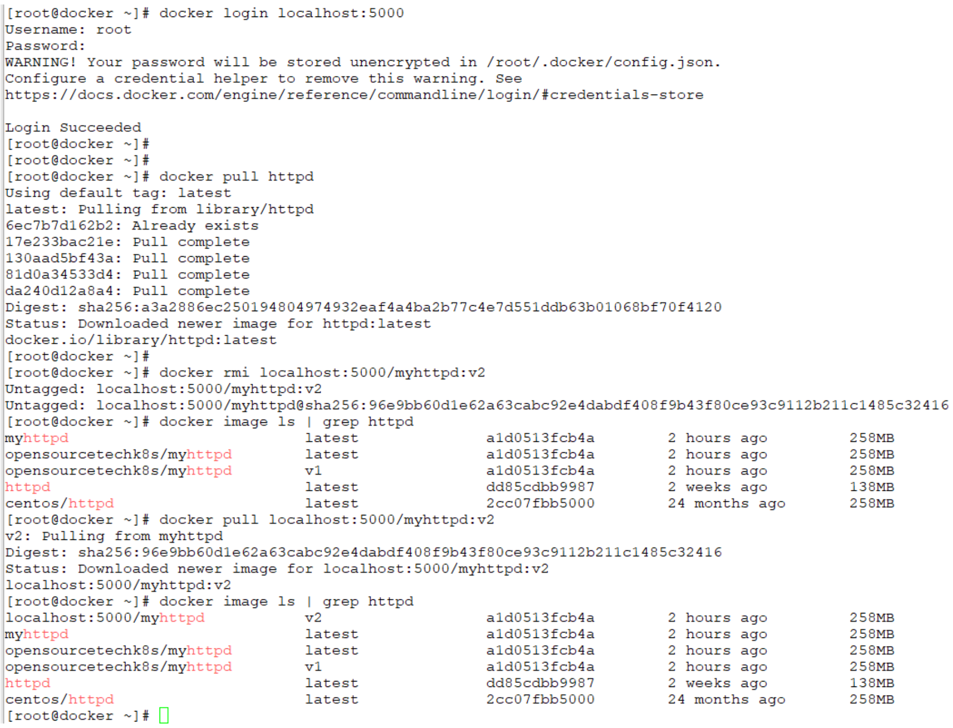
# docker run -d -p <*host port:container port*> --name <*Container Name*> <*Image Name*>

- Host port = 5000
- Container port = 5000
- Container Name = local_registry
- Image Name = registry (not available locally so downloading from Docker HUB)



```
[root@docker ~]# docker run -d -p 5000:5000 --name local-repostry registry
9fa56f65a44da5e6d9251ed679b42e00bf8e8f8972c08f23f7764934f2e1a888
[root@docker ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
9fa56f65a44d        registry            "/entrypoint.sh /etc…"   25 seconds ago      Up 24 seconds       0.0.0.0:5000->5000/tcp   local-repostry
[root@docker ~]# docker tag opensourcetechk8s/myhttpd:v1 localhost:5000/myhttpd:v2
[root@docker ~]# docker image ls | grep myhttpd
localhost:5000/myhttpd            v2                  a1d0513fcb4a        About an hour ago   258MB
myhttpd                           latest              a1d0513fcb4a        About an hour ago   258MB
opensourcetechk8s/myhttpd         latest              a1d0513fcb4a        About an hour ago   258MB
opensourcetechk8s/myhttpd         v1                  a1d0513fcb4a        About an hour ago   258MB
[root@docker ~]# docker push localhost:5000/myhttpd
The push refers to repository [localhost:5000/myhttpd]
9351939f70b8: Pushed
920640105caf: Pushed
7c937d8a9f4f: Pushed
d15c61d3ecda: Pushed
071d8bd76517: Pushed
v2: digest: sha256:96e9bb60d1e62a63cabc92e4dabdf408f9b43f80ce93c9112b211c1485c32416 size: 1363
[root@docker ~]# 
```

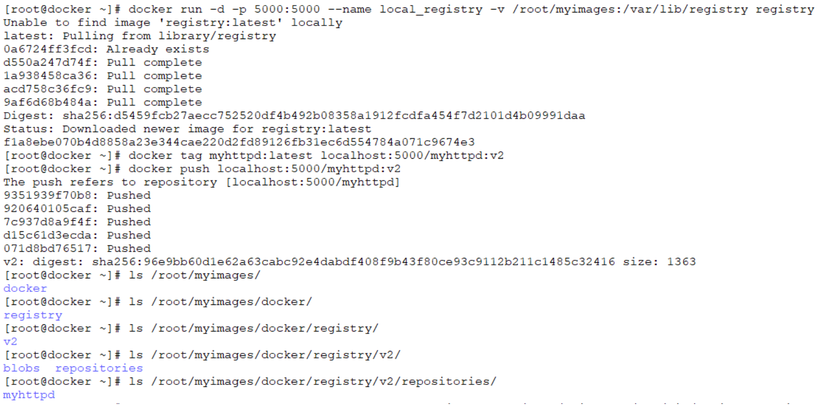
Browse registry with your docker host ip address : http://192.168.1.79:5000/v2/_catalog



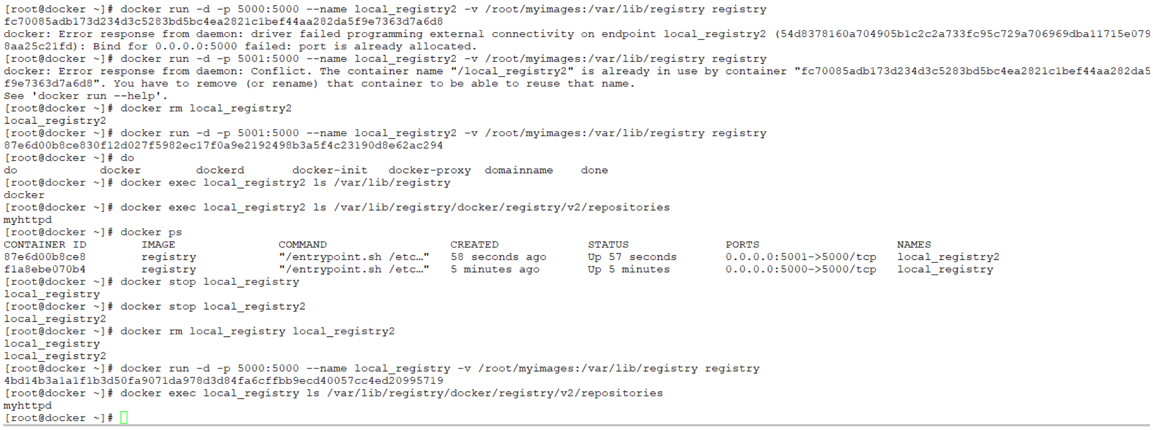
```
[root@docker ~]# docker login localhost:5000
Username: root
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[root@docker ~]#
[root@docker ~]#
[root@docker ~]# docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
6ec7b7d162b2: Already exists
17e233bac21e: Pull complete
130aad5bf43a: Pull complete
81d0a34533d4: Pull complete
da240d12a8a4: Pull complete
Digest: sha256:a3a2886ec250194804974932eaf4a4ba2b77c4e7d551ddb63b01068bf70f4120
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
[root@docker ~]#
[root@docker ~]# docker rmi localhost:5000/myhttpd:v2
Untagged: localhost:5000/myhttpd:v2
Untagged: localhost:5000/myhttpd@sha256:96e9bb60d1e62a63cabc92e4dabdf408f9b43f80ce93c9112b211c1485c32416
[root@docker ~]# docker image ls | grep httpd
myhttpd                         latest              a1d0513fcb4a        2 hours ago         258MB
opensourcetechk8s/myhttpd       latest              a1d0513fcb4a        2 hours ago         258MB
opensourcetechk8s/myhttpd       v1                  a1d0513fcb4a        2 hours ago         258MB
httpd                           latest              dd85cdbb9987        2 weeks ago         138MB
centos/httpd                    latest              2cc07fbb5000        24 months ago       258MB
[root@docker ~]# docker pull localhost:5000/myhttpd:v2
v2: Pulling from myhttpd
Digest: sha256:96e9bb60d1e62a63cabc92e4dabdf408f9b43f80ce93c9112b211c1485c32416
Status: Downloaded newer image for localhost:5000/myhttpd:v2
localhost:5000/myhttpd:v2
[root@docker ~]# docker image ls | grep httpd
localhost:5000/myhttpd          v2                  a1d0513fcb4a        2 hours ago         258MB
myhttpd                         latest              a1d0513fcb4a        2 hours ago         258MB
opensourcetechk8s/myhttpd       latest              a1d0513fcb4a        2 hours ago         258MB
opensourcetechk8s/myhttpd       v1                  a1d0513fcb4a        2 hours ago         258MB
httpd                           latest              dd85cdbb9987        2 weeks ago         138MB
centos/httpd                    latest              2cc07fbb5000        24 months ago       258MB
[root@docker ~]# 
```

Persistent Disk

- For docker registry



```
[root@docker ~]# docker run -d -p 5000:5000 --name local_registry -v /root/myimages:/var/lib/registry registry
Unable to find image 'registry:latest' locally
latest: Pulling from library/registry
0a6724ff3fcd: Already exists
d550a247d74f: Pull complete
1a938458ca36: Pull complete
acd758c36fc9: Pull complete
9af6d68b484a: Pull complete
Digest: sha256:d5459fcb27aecc752520df4b492b08358a1912fcdfa454f7d2101d4b09991daa
Status: Downloaded newer image for registry:latest
f1a8ebe070b4d8858a23e344cae220d2fd89126fb31ec6d554784a071c9674e3
[root@docker ~]# docker tag myhttpd:latest localhost:5000/myhttpd:v2
[root@docker ~]# docker push localhost:5000/myhttpd:v2
The push refers to repository [localhost:5000/myhttpd]
9351939f70b8: Pushed
920640105caf: Pushed
7c937d8a9f4f: Pushed
d15c61d3ecda: Pushed
071d8bd76517: Pushed
v2: digest: sha256:96e9bb60d1e62a63cabc92e4dabdf408f9b43f80ce93c9112b211c1485c32416 size: 1363
[root@docker ~]# ls /root/myimages/
docker
[root@docker ~]# ls /root/myimages/docker/
registry
[root@docker ~]# ls /root/myimages/docker/registry/
v2
[root@docker ~]# ls /root/myimages/docker/registry/v2/
blobs  repositories
[root@docker ~]# ls /root/myimages/docker/registry/v2/repositories/
myhttpd
```

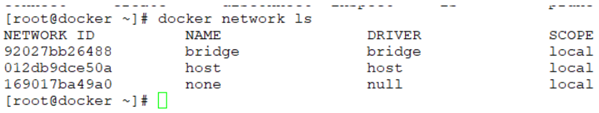


```
[root@docker ~]# docker run -d -p 5000:5000 --name local_registry2 -v /root/myimages:/var/lib/registry registry
fc70085adb173d234d3c5283bd5bc4ea2821c1bef44aa282da5f9e7363d7a6d8
docker: Error response from daemon: driver failed programming external connectivity on endpoint local_registry2 (54d8378160a704905b1c2c2a733fc95c729a706969dba11715e079
8aa25c21fd): Bind for 0.0.0.0:5000 failed: port is already allocated.
[root@docker ~]# docker run -d -p 5001:5000 --name local_registry2 -v /root/myimages:/var/lib/registry registry
docker: Error response from daemon: Conflict. The container name "/local_registry2" is already in use by container "fc70085adb173d234d3c5283bd5bc4ea2821c1bef44aa282da5
f9e7363d7a6d8". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
[root@docker ~]# docker rm local_registry2
local_registry2
[root@docker ~]# docker run -d -p 5001:5000 --name local_registry2 -v /root/myimages:/var/lib/registry registry
87e6d00b8ce830f12d027f5982ec17f0a9e2192498b3a5f4c23190d8e62ac294
[root@docker ~]# do
do              docker          dockerd         docker-init     docker-proxy  domainname     done
[root@docker ~]# docker exec local_registry2 ls /var/lib/registry
docker
[root@docker ~]# docker exec local_registry2 ls /var/lib/registry/docker/registry/v2/repositories
myhttpd
[root@docker ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
87e6d00b8ce8        registry            "/entrypoint.sh /etc…"   58 seconds ago      Up 57 seconds       0.0.0.0:5001->5000/tcp   local_registry2
f1a8ebe070b4        registry            "/entrypoint.sh /etc…"   5 minutes ago       Up 5 minutes        0.0.0.0:5000->5000/tcp   local_registry
[root@docker ~]# docker stop local_registry
local_registry
[root@docker ~]# docker stop local_registry2
local_registry2
[root@docker ~]# docker rm local_registry local_registry2
local_registry
local_registry2
[root@docker ~]# docker run -d -p 5000:5000 --name local_registry -v /root/myimages:/var/lib/registry registry
4bd14b3a1a1f1b3d50fa9071da978d3d84fa6cffbb9ecd40057cc4ed20995719
[root@docker ~]# docker exec local_registry ls /var/lib/registry/docker/registry/v2/repositories
myhttpd
[root@docker ~]# 
```

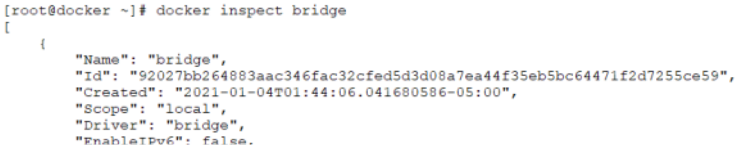
Networking

- Bridge
- Network

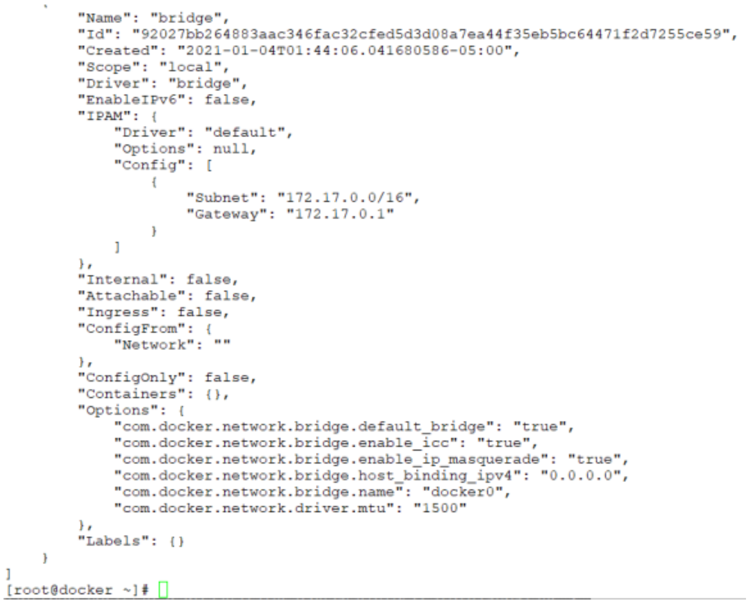
- **List available images and Network drivers**



```
[root@docker ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
92027bb26488        bridge              bridge              local
012db9dce50a        host                host                local
169017ba49a0        none                null                local
[root@docker ~]# 
```

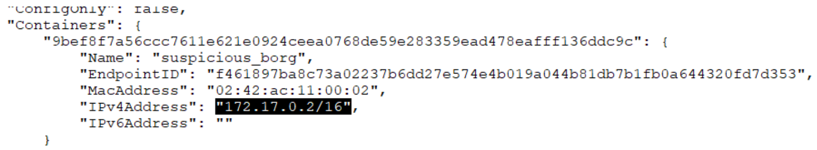


```
[root@docker ~]# docker inspect bridge
[
    {
        "Name": "bridge",
        "Id": "92027bb264883aac346fac32cfed5d3d08a7ea44f35eb5bc64471f2d7255ce59",
        "Created": "2021-01-04T01:44:06.041680586-05:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
```



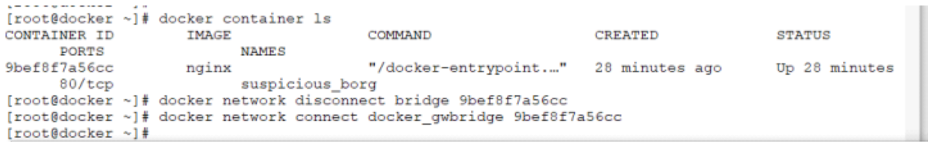
```
        "Name": "bridge",
        "Id": "92027bb264883aac346fac32cfed5d3d08a7ea44f35eb5bc64471f2d7255ce59",
        "Created": "2021-01-04T01:44:06.041680586-05:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
[root@docker ~]# 
```

- **Default network driver is Bridge.**

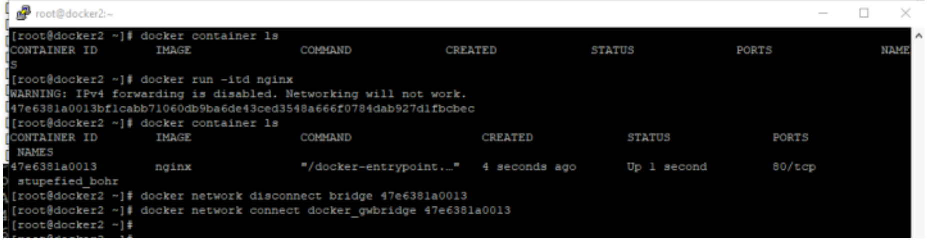


```
"ConfigOnly": false,
"Containers": {
    "9bef8f7a56ccc7611e621e0924ceea0768de59e283359ead478eafff136ddc9c": {
        "Name": "suspicious_borg",
        "EndpointID": "f461897ba8c73a02237b6dd27e574e4b019a044b81db7b1fb0a644320fd7d353",
        "MacAddress": "02:42:ac:11:00:02",
        "IPv4Address": "172.17.0.2/16",
        "IPv6Address": ""
    }
```

- **Connect standalone containers running on two different docker hosts:**



```
[root@docker ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
      PORTS               NAMES
9bef8f7a56cc        nginx               "/docker-entrypoint.…"   28 minutes ago      Up 28 minutes
      80/tcp              suspicious_borg
[root@docker ~]# docker network disconnect bridge 9bef8f7a56cc
[root@docker ~]# docker network connect docker_gwbridge 9bef8f7a56cc
[root@docker ~]#
```



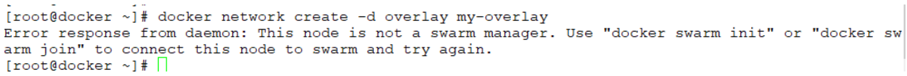
root@docker2:~

```
[root@docker2 ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAME
S
[root@docker2 ~]# docker run -itd nginx
WARNING: IPv4 forwarding is disabled. Networking will not work.
47e6381a0013bf1cabb71060db9ba6de43ced3548a666f0784dab927d1fbcbec
[root@docker2 ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
 NAMES
47e6381a0013        nginx               "/docker-entrypoint.…"   4 seconds ago       Up 1 second         80/tcp
 stupefied_bohr
[root@docker2 ~]# docker network disconnect bridge 47e6381a0013
[root@docker2 ~]# docker network connect docker_gwbridge 47e6381a0013
[root@docker2 ~]#
```

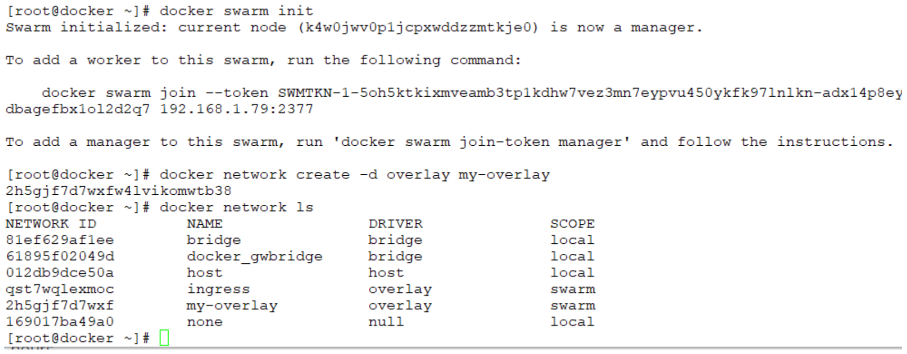
NOTE:- By default docker bridge doesn't allow containers to ping each other, for the same we have to create custom (user -defined) bridge.

Overlay: Used by SWARM

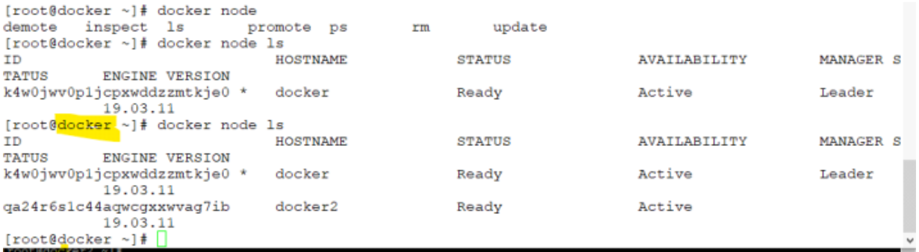
- To create an overlay network for use with swarm services, use a command like the following:



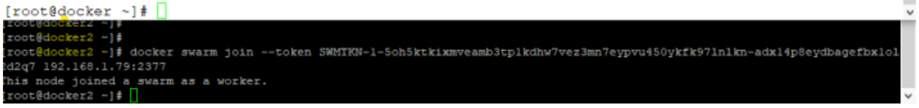
```
[root@docker ~]# docker network create -d overlay my-overlay
Error response from daemon: This node is not a swarm manager. Use "docker swarm init" or "docker sw
arm join" to connect this node to swarm and try again.
[root@docker ~]# 
```



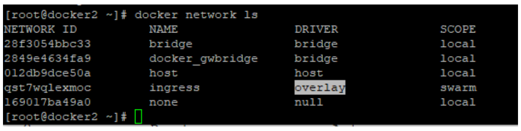
```
[root@docker ~]# docker swarm init
Swarm initialized: current node (k4w0jwv0p1jcpxwddzmtkje0) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-5oh5ktkixmveamb3tp1kdhw7vez3mn7eypvu450ykfk971nlkn-adx14p8ey
dbagefbx1o12d2q7 192.168.1.79:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

[root@docker ~]# docker network create -d overlay my-overlay
2h5gjf7d7wxfw4lvikomwtb38
[root@docker ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
81ef629af1ee        bridge              bridge              local
61895f02049d        docker_gwbridge     bridge              local
012db9dce50a        host                host                local
qst7wqlexmoc        ingress             overlay             swarm
2h5gjf7d7wxf        my-overlay          overlay             swarm
169017ba49a0        none                null                local
[root@docker ~]# 
```



```
[root@docker ~]# docker node
demote    inspect  ls        promote  ps        rm        update
[root@docker ~]# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER S
TATUS      ENGINE VERSION
k4w0jwv0p1jcpxwddzzmtkje0 *   docker              Ready               Active              Leader
           19.03.11
[root@docker ~]# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER S
TATUS      ENGINE VERSION
k4w0jwv0p1jcpxwddzzmtkje0 *   docker              Ready               Active              Leader
           19.03.11
qa24r6s1c44aqwcgxxwvag7ib     docker2             Ready               Active
           19.03.11
[root@docker ~]# 
```



```
[root@docker2 ~]#
[root@docker2 ~]#
[root@docker2 ~]# docker swarm join --token SWMTKN-1-5oh5ktkixmveamb3tplkdhw7vez3mn7eypvu450ykfk971nlkn-adxl4p8eydbagefbxlol
d2q7 192.168.1.79:2377
This node joined a swarm as a worker.
[root@docker2 ~]# 
```



```
[root@docker2 ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
28f3054bbc33        bridge              bridge              local
2849e4634fa9        docker_gwbridge     bridge              local
012db9dce50a        host                host                local
qst7wqlexmoc        ingress             overlay             swarm
169017ba49a0        none                null                local
[root@docker2 ~]# 
```