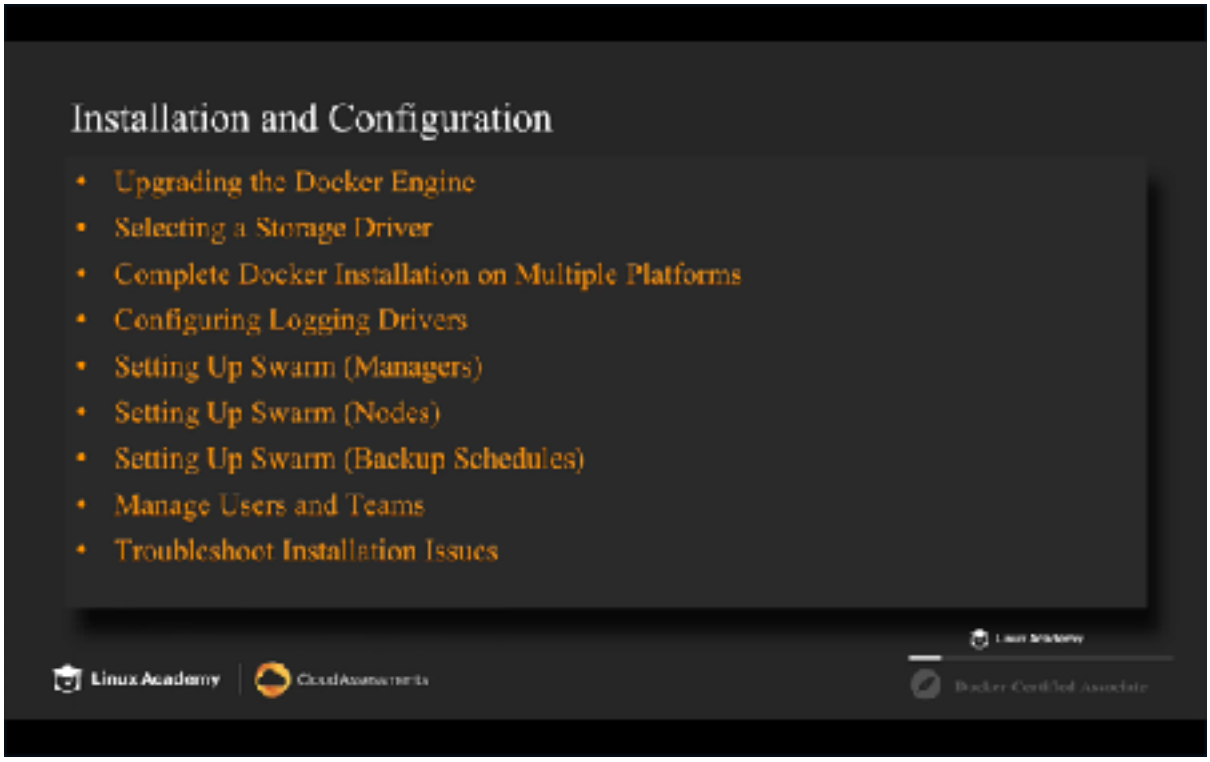
Docker – Linux Academy

```
cloud_user@bakumar1c ~]$ sudo -i
[sudo] password for cloud_user:
[root@bakumar1c ~]#
```

Agenda



Installation and Configuration (Continued)

- Sizing Your Environment
- Namespaces, CGroups and Certificates
- Authenticate Docker Daemon with Certificates
- Configuring Backups

Image Creation, Management and Registry

- Understanding Dockerfile Options
- Dockerfile Structure
- Image Efficiencies in Dockerfile
- Manage Docker Images at the CLI
- Tag Images
- Using Registries for Image Storage
- Understanding Docker Image Layering
- Apply Files to Create Docker Images
- Modifying a Single Image Layer

Image Creation, Management and Registry (Continued)

- Deploy an Image Repository
- Configure an Image Repository for Use
- Searching an Image Repository
- Managing Images in Your Repository
- Deleting Images

Orchestration

- Setup a Swarm Cluster
- Containers vs. Services
- Locking a Swarm Cluster
- Creating a Swarm Service from a Container
- Understanding the 'docker inspect' Output
- Creating a Stack File Using YAML with 'docker stack deploy'
- Working with a Running Service Stack
- Increase and Decrease Replica Numbers
- Working with Networks and Ports

Orchestration (Continued)

- Working with Volumes
- Running Replicated vs. Global Services
- Troubleshoot Service Deployments
- Apply Node Labels for Task Placement
- How Dockerized Apps Communicate with Legacy Systems
- Why Quorum is Important in a Swarm Cluster
- Using Templates with 'docker service create'

Storage and Volumes

- Graph Drivers on Multiple Platforms
- Configuring Devicemapper
- Object Storage vs. Block Storage
- Application Layers and the Filesystem
- Using Volumes for Persistent Storage
- Cleaning Up Images on Filesystems and on DTR
- Using Storage Across Cluster Nodes

Networking

- Creating Docker Bridge Networks
- Troubleshooting Container Connectivity Issues
- Exposing Ports for External Access
- Identify IPs for Internal and External Container Interfaces
- Understanding Built-In Network Drivers
- Understanding the Container Network Model
- Configure Docker for External DNS
- Using Docker to Load Balance Web Traffic

Networking (Continued)

- Traffic Flow Between Docker Components
- Services on the Docker Overlay Network
- Host vs. Ingress Port Publishes Modes

Security

- Signing Docker Images
- Complete a Security Scan on an Image
- Working with Docker Content Trust
- Configure RBAC in UCP
- Integrate UCP with LDAP/AD
- UCP Client Bundle Creation
- Docker Swarm and Docker Engine Security
- MTLS
- Identity Policies
- UCP Workers vs. Managers
- Use External Certificates with UCP and DTR

Docker Certified Associate

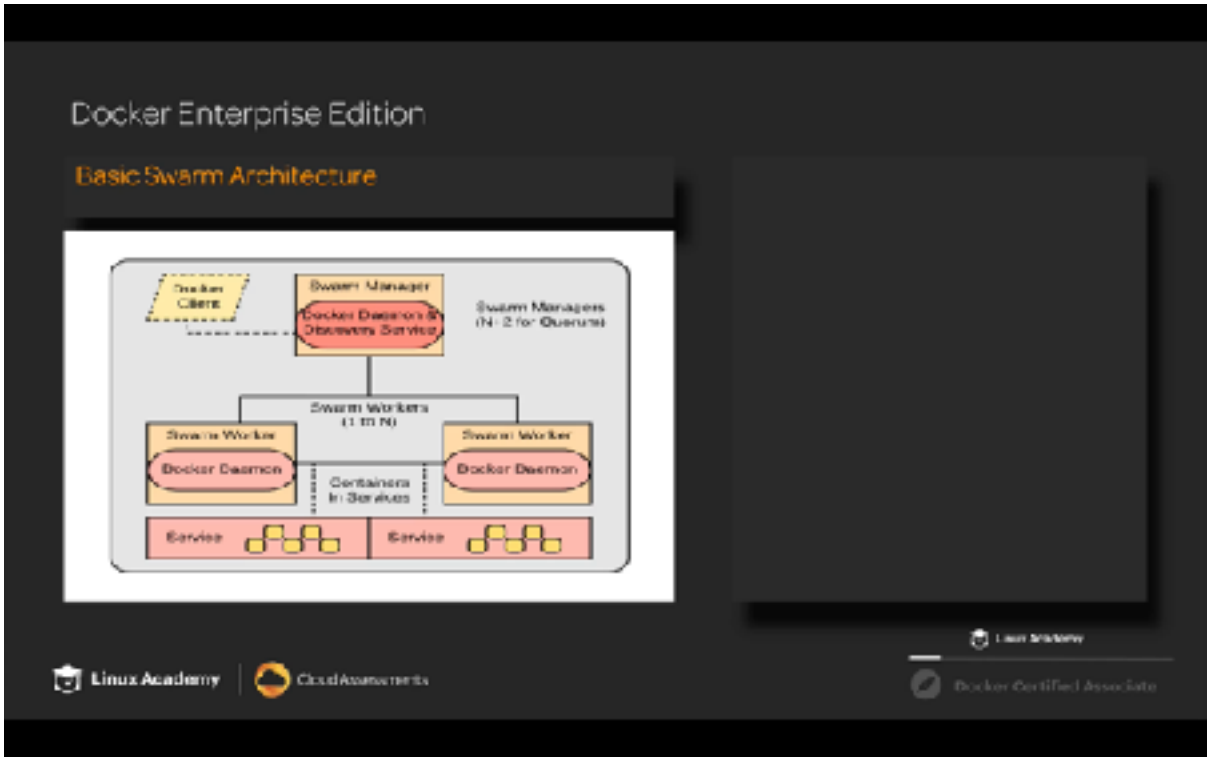
Docker Swarm and Enterprise Edition

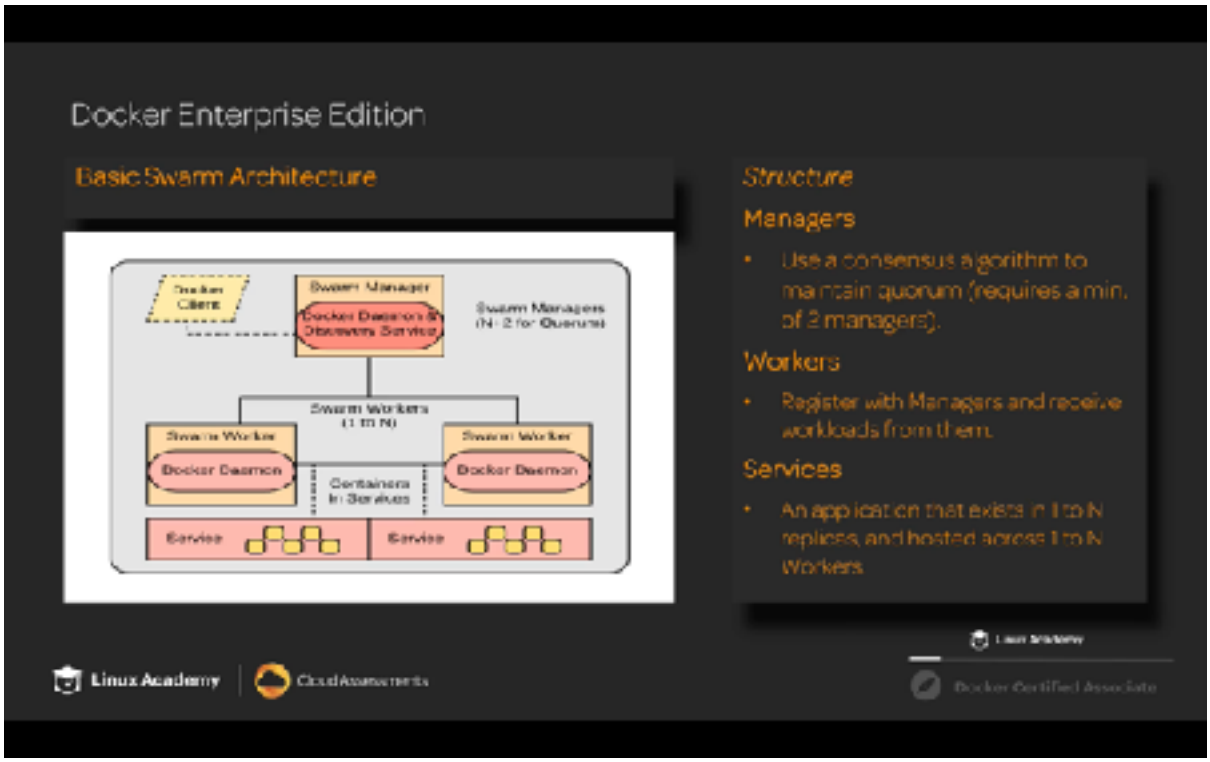
Docker Swarm

Available in the Docker Enterprise Edition (at all tiers), as well as the Docker Community Edition, Docker Swarm is a clustering and scheduling tool for clusters of Docker containers (grouped together as services).

Swarms allow portability, abstraction, flexibility, and consistency of complex application service deployments on any supported infrastructure.

Docker Swarm Manager(s) are responsible for validating, logging the state of, and distributing instructions to Docker Swarm Workers. The Docker service (daemon) is installed on every node in a Swarm (Manager or Worker).





Docker installation

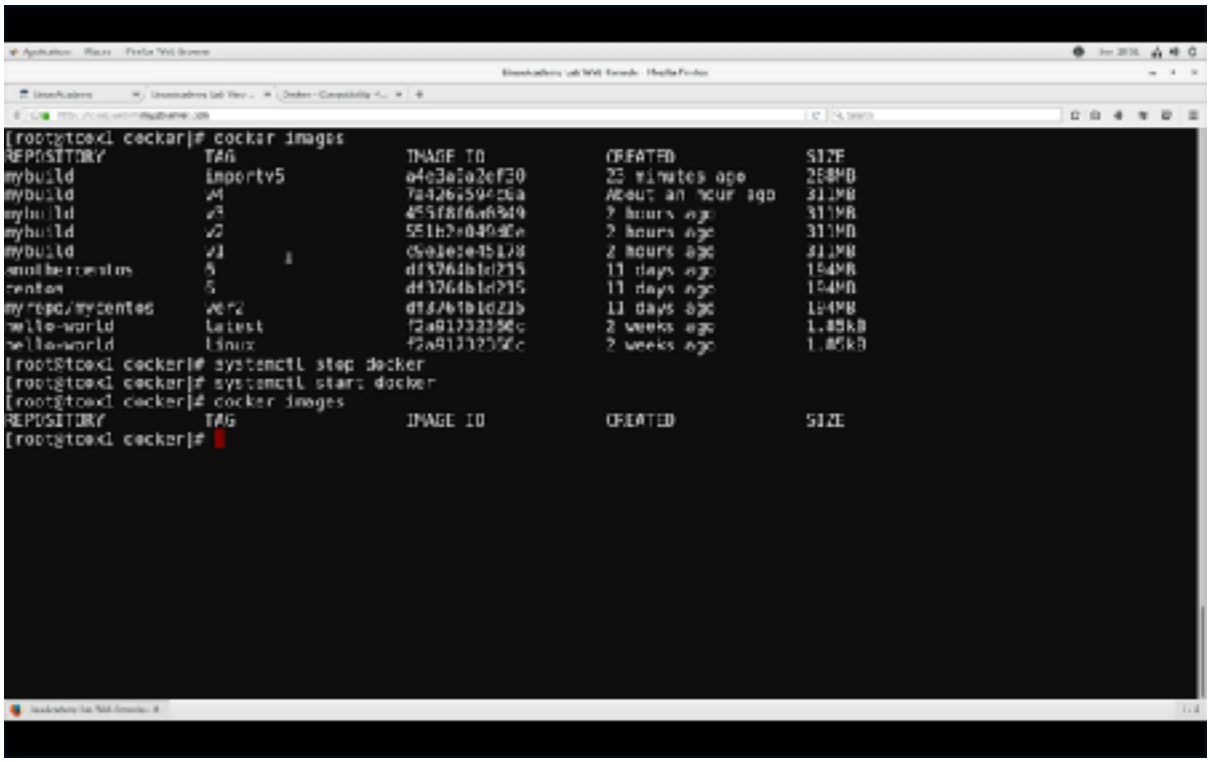
```
[root@bakumar1c ~]# yum install -y yum-utils device-mapper-persistent-data lvm2

Repo adding:
#yum-config-manager  --add-repo https://download.docker.com/linux/centos/docker-
ce.repo
 #yum update
Insatlling docker-ce version which is officially supported
#yum install docker-ce
#systemctl  enable docker && systemctl  start docker && systemctl  status docker
```

```
[root@bakumar1c ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
[root@bakumar1c ~]# docker ps
CONTAINER ID        IMAGE
```

```
[user@tcox1 ~]$ docker images
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get ht
tp://%2Fvar%2Frun%2Fdocker.sock/v1.32/images/json: dial unix /var/run/docker.sock: connect: permission denied
[user@tcox1 ~]$ sudo su -
[sudo] password for user:
Last login: Thu Dec  7 01:28:01 UTC 2017 on pts/0
[root@tcox1 ~]# cd /var/run
[root@tcox1 run]# ls -al docker.sock
srw-rw----. 1 root docker 0 Dec  7 01:51 docker.sock
[root@tcox1 run]# usermod -aG docker user
[root@tcox1 run]# exit
logout
[user@tcox1 ~]$ docker images
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get ht
tp://%2Fvar%2Frun%2Fdocker.sock/v1.32/images/json: dial unix /var/run/docker.sock: connect: permission denied
[user@tcox1 ~]$ exit
logout
Connection to tcox1.mylabserver.com closed.
[tcox@ca-fedoravm ~]$ ssh user@tcox1.mylabserver.com
user@tcox1.mylabserver.com's password:
```

```
[user@tcox1 ~]$ docker images
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get ht
tp://%2Fvar%2Frun%2Fdocker.sock/v1.32/images/json: dial unix /var/run/docker.sock: connect: permission denied
[user@tcox1 ~]$ sudo su -
[sudo] password for user:
Last login: Thu Dec  7 01:28:01 UTC 2017 on pts/0
[root@tcox1 ~]# cd /var/run
[root@tcox1 run]# ls -al docker.sock
srw-rw----. 1 root docker 0 Dec  7 01:51 docker.sock
[root@tcox1 run]# usermod -aG docker user
[root@tcox1 run]# exit
logout
[user@tcox1 ~]$ docker images
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get ht
tp://%2Fvar%2Frun%2Fdocker.sock/v1.32/images/json: dial unix /var/run/docker.sock: connect: permission denied
[user@tcox1 ~]$ exit
logout
Connection to tcox1.mylabserver.com closed.
[tcox@ca-fedoravm ~]$ ssh user@tcox1.mylabserver.com
user@tcox1.mylabserver.com's password:
Last login: Thu Dec  7 01:27:42 2017 from 136.63.34.144
[user@tcox1 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
[user@tcox1 ~]$
```

Docker Daemon


[root@tcex1 cocker]# cocker images
REPOSITORY TAG IMAGE ID CREATED SIZE
mybuild importv5 a4e3a1a2ef30 23 minutes ago 258MB
mybuild v4 7a426c594c6a About an hour ago 311MB
mybuild v3 455f816a8a49 2 hours ago 311MB
mybuild v2 551b2e049d2e 2 hours ago 311MB
mybuild v1 c5e1e1e45178 2 hours ago 311MB
anotherrcentos 5 d1376b1d215 11 days ago 194MB
centos 5 d1376b1d215 11 days ago 194MB
myrepo/mycentos v5.2 d1376b1d215 11 days ago 194MB
hello-world latest f2a91732366c 2 weeks ago 1.85kB
hello-world linux f2a91732366c 2 weeks ago 1.85kB
[root@tcex1 cocker]# systemctl stop docker
[root@tcex1 cocker]# systemctl start docker
[root@tcex1 cocker]# cocker images
REPOSITORY TAG IMAGE ID CREATED SIZE
[root@tcex1 cocker]#

```
hello-world         latest              f2a91732366c        2 weeks ago         1.85kB
[root@tcxl docker]# systemctl stop docker
[root@tcxl docker]# systemctl start docker
[root@tcxl docker]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
[root@tcxl docker]# cd /var/lib/docker
[root@tcxl docker]# ll
total 0
drwx------. 2 root root   29 Dec  7 01:51 builder
drwx------. 4 root root   72 Dec 11 00:24 containers
drwx------. 4 root root   40 Dec 11 00:51 devicemapper
drwx------. 4 root root   39 Dec 11 00:51 image
drwxr-x---. 3 root root   18 Dec  7 01:51 network
drwx------. 5 root root 4096 Dec 11 00:26 overlay
drwx------. 4 root root   38 Dec  7 01:51 plugins
drwx------. 2 root root    6 Dec  7 01:51 swarm
drwx------. 2 root root    6 Dec 11 00:51 tmp
drwx------. 2 root root    6 Dec  7 01:51 trust
drwx------. 5 root root 4096 Dec 11 00:24 volumes
[root@tcxl docker]# cd devicemapper
[root@tcxl devicemapper]# ll
total 0
drwx------. 2 root root 32 Dec 11 00:51 devicemapper
drwx------. 2 root root 63 Dec 11 00:51 metadata
[root@tcxl devicemapper]#  cd devicemapper/
[root@tcxl devicemapper]# ll
total 11340
-rw-------. 1 root root 107374182400 Dec 11 00:51 data
-rw-------. 1 root root   2147483648 Dec 11 00:51 metadata
[root@tcxl devicemapper]#
```

```
[root@tcxl ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              5                   d13764b1d235        11 days ago         124MB
[root@tcxl ~]# docker info | grep Storage
WARNING: devicemapper: usage of loopback devices is strongly discouraged for production use.
         Use `--storage-opt dm.thinpooldev` to specify a custom block storage device.
Storage Driver: devicemapper
[root@tcxl ~]#
```

Docker info

```
# docker info
Client:
 Debug Mode: false

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 19.03.3
 Storage Driver: devicemapper
  Pool Name: docker-259:1-17567229-pool
  Pool Blocksize: 65.54kB
  Base Device Size: 10.74GB
  Backing Filesystem: xfs
  Udev Sync Supported: true
  Data file: /dev/loop0
  Metadata file: /dev/loop1
  Data loop file: /var/lib/docker/devicemapper/devicemapper/data
  Metadata loop file: /var/lib/docker/devicemapper/devicemapper/metadata
  Data Space Used: 11.73MB
  Data Space Total: 107.4GB
  Data Space Available: 13.82GB
  Metadata Space Used: 17.36MB
  Metadata Space Total: 2.147GB
  Metadata Space Available: 2.13GB
  Thin Pool Minimum Free Space: 10.74GB
  Deferred Removal Enabled: true
  Deferred Deletion Enabled: true
  Deferred Deleted Device Count: 0
  Library Version: 1.02.158-RHEL7 (2019-05-13)
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 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: b34a5c8af56e510852c35414db4c1f4fa6172339
 runc version: 3e425f80a8c931f88e6d94a8c831b9d5aa481657
 init version: fec3683
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 3.10.0-1062.1.1.el7.x86_64
 Operating System: CentOS Linux 7 (Core)
 OSType: linux
 Architecture: x86_64
 CPUs: 2
 Total Memory: 1.758GiB
 Name: bakumar2c.mylabserver.com
 ID: ZDH2:6XCI:B7TG:R4LA:7PXF:Z5AT:5ZCN:CGX3:T63X:3HCE:BV32:5SJL
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Registry: https://index.docker.io/v1/
 Labels:
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false

WARNING: the devicemapper storage-driver is deprecated, and will be removed in a future release.
WARNING: devicemapper: usage of loopback devices is strongly discouraged for production use.
         Use `--storage-opt dm.thinpooldev` to specify a custom block storage device.
[root@bakumar2c ~]#
```

Docker default logging

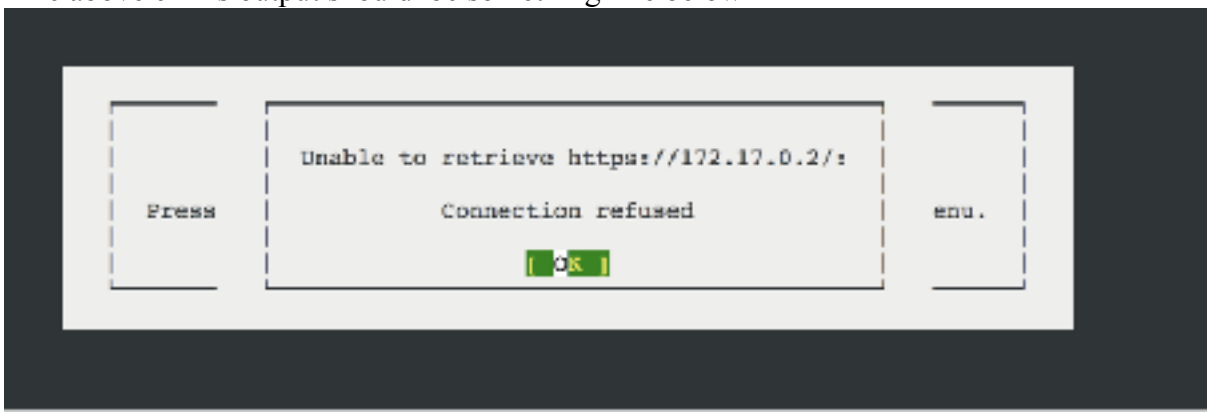
```
# docker info | grep Logging
WARNING: the devicemapper storage-driver is deprecated, and will be removed in a future release.
WARNING: devicemapper: usage of loopback devices is strongly discouraged for production use.
        Use `--storage-opt dm.thinpooldev` to specify a custom block storage device.
 Logging Driver: json-file
```

```
   26  docker image pull httpd
   27  docker images
   28  docker ps

[root@bakumar1c devicemapper]# docker pull centos:6
6: Pulling from library/centos
ff50d722b382: Pull complete
Digest: sha256:dec8f471302de43f4cfcf82f56d99a5227b5ea1aa6d02fa56344986e1f4610e7
Status: Downloaded newer image for centos:6
docker.io/library/centos:6
 [root@bakumar1c devicemapper]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              6                   d0957ffdf8a2        7 months ago        194MB
 [root@bakumar1c devicemapper]# docker info | grep Storage
WARNING: the devicemapper storage-driver is deprecated, and will be removed in a future release.
WARNING: devicemapper: usage of loopback devices is strongly discouraged for production use.
        Use `--storage-opt dm.thinpooldev` to specify a custom block storage device.
 Storage Driver: devicemapper
[root@bakumar1c devicemapper]#    30  docker ps
   31  docker container inspect testweb
   32  docker container inspect testweb | grep IPAdd
   36  /bin/yum provides telnet
   37  /bin/yum install -y  telnet elinks
   38  docker container inspect testweb | grep IPAdd
   39  elinks 172.17.0.2
```

Logging driver

```
[root@bakumar1c devicemapper]# docker info | grep Logging
 Logging Driver: json-file
WARNING: the devicemapper storage-driver is deprecated, and will be removed in a future release.
WARNING: devicemapper: usage of loopback devices is strongly discouraged for production use.
        Use `--storage-opt dm.thinpooldev` to specify a custom block storage device.
[root@bakumar1c devicemapper]#


[root@bakumar1c devicemapper]# docker image pull httpd
Using default tag: latest
 [root@bakumar1c devicemapper]# docker container run -d --name testApp httpd
 [root@bakumar1c devicemapper]# docker ps
CONTAINER ID    IMAGE     COMMAND            CREATED          STATUS          PORTS     NAMES
74dde9c86011    httpd     "httpd-foreground"  14 seconds ago   Up 3 seconds    80/tcp    testApp

 [root@bakumar1c devicemapper]# docker container inspect testApp info
[
   {< huge set data >
}
]




# docker container inspect  testApp | grep -i ipaddr
            "SecondaryIPAddresses": null,
            "IPAddress": "172.17.0.2",
                  "IPAddress": "172.17.0.2",
[root@bakumar1c devicemapper]# elinks https://172.17.0.2
```

The above elinks output should be something like below



Docker logs

```
[root@bakumar1c devicemapper]# docker logs testApp
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
[Sat Oct 12 06:15:02.032652 2019] [mpm_event:notice] [pid 1:tid 140231771628672] AH00489: Apache/2.4.41 (Unix) configured -- resuming normal operations
[Sat Oct 12 06:15:02.364632 2019] [core:notice] [pid 1:tid 140231771628672] AH00094: Command line: 'httpd -D FOREGROUND'
 [root@bakumar1c devicemapper]# docker stop testApp
testApp
[root@bakumar1c devicemapper]# docker rm testApp
testApp
[root@bakumar1c devicemapper]#
 [root@bakumar1c devicemapper]# cd /etc/docker/
[root@bakumar1c docker]# ll
total 8
-rw-r--r--. 1 root root  99 Oct 12 05:55 deamon.json
-rw-------. 1 root root 244 Oct 11 19:05 key.json
[root@bakumar1c docker]# docker info | grep -i logging
WARNING: the devicemapper storage-driver is deprecated, and will be removed in a future release.
WARNING: devicemapper: usage of loopback devices is strongly discouraged for production use.
        Use `--storage-opt dm.thinpooldev` to specify a custom block storage device.
 Logging Driver: json-file
[root@bakumar1c docker]#
[root@bakumar1c docker]#
[root@bakumar1c docker]#
[root@bakumar1c docker]# vim /etc/rsyslog.conf
[root@bakumar1c docker]# cat /etc/docker/deamon.json

{
  "log-driver": "syslog",
  "log-opts": {
    "syslog-address": "udp://172.31.42.108:514"
  }
}
[root@bakumar1c docker]# systemctl  start rsyslog
[root@bakumar1c docker]# systemctl  status rsyslog


[root@bakumar1c docker]# systemctl  start docker
[root@bakumar1c docker]# systemctl  status docker


[root@bakumar1c docker]# docker info | grep -i logging
WARNING: the devicemapper storage-driver is deprecated, and will be removed in a future release.
WARNING: devicemapper: usage of loopback devices is strongly discouraged for production use.
        Use `--storage-opt dm.thinpooldev` to specify a custom block storage device.
 Logging Driver: json-file
[root@bakumar1c docker]#
```

But still its pointing to json-file bit not syslog

```
[root@bakumar1c docker]# docker container run -d --name testSyslog --log-driver syslog
httpd
a74d74428e39e4493a074aec1b49943c39f82fa8dbe37291d06a9e4bdbb2783c
[root@bakumar1c docker]#
[root@bakumar1c docker]#
```

Dcoker Swarm

docker swarm init --advertise-addr 172.31.42.108
Swarm initialized: current node (3b4nswbkipb25ir9f7ynbatlx) is now a manager.

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

docker swarm join --token SWMTKN-1-2y2kyca449o661v9e4x7tgti4nz9bpyr5ktl1f2cnjqlpapwp6-90e9782vkg4d5xwwzesnso2r1 172.31.42.108:2377

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

If we lost above token, we can collect like below

Manager token
#docker swarm join-token manager
To add a manager to this swarm, run the following command:
docker swarm join --token SWMTKN-1-2y2kyca449o661v9e4x7tgti4nz9bpyr5ktl1f2cnjqlpapwp6-cmauhmmlia3hrtvbh88o9a0s8 172.31.42.108:2377

Worker Token: To join a Worker node in swarm, run below
docker swarm join-token worker
To add a worker to this swarm, run the following command:
docker swarm join --token SWMTKN-1-2y2kyca449o661v9e4x7tgti4nz9bpyr5ktl1f2cnjqlpapwp6-90e9782vkg4d5xwwzesnso2r1 172.31.42.108:2377

to join another **manager** in the **swarm**, run below command

[root@bakumar1c docker]# docker swarm join-token manager
To add a manager to this swarm, run the following command:

docker swarm join --token SWMTKN-1-2y2kyca449o661v9e4x7tgti4nz9bpyr5ktl1f2cnjqlpapwp6-cmauhmmlia3hrtvbh88o9a0s8 172.31.42.108:2377

Build two more nodes:
Node2:
[root@bakumar2c ~]# hostname
bakumar2c.mylabserver.com
Node 3:
[root@bakumar3c ~]# hostname
bakumar3c.mylabserver.com

manager Node: where Swarm manager is running

```
[root@bakumar1c docker]# docker node ls
ID                    HOSTNAME                STATUS    AVAILABILITY    MANAGER STATUS    ENGINE VERSION
3b4nswbkipb25ir9f7ynbatlx *   bakumar1c.mylabserver.com   Ready    Active    Leader    19.03.3
[root@bakumar1c docker]#
bakumar1c.mylabserver.com
[root@bakumar1c docker]# hostname
```

Adding worker nodes using join-token

From Node2:

```
[root@bakumar2c ~]#    docker swarm join --token
SWMTKN-1-2y2kyca449o661v9e4x7tgti4nz9bpyr5ktl1f2cnjqlpapwp6-90e9782vkg4d5xwwzesnso2r1
172.31.42.108:2377
This node joined a swarm as a worker.
```

After adding, from Manager node, list of nodes:

```
[root@bakumar1c docker]# docker node ls
ID                    HOSTNAME                STATUS    AVAILABILITY    MANAGER STATUS    ENGINE VERSION
3b4nswbkipb25ir9f7ynbatlx *   bakumar1c.mylabserver.com   Ready    Active    Leader    19.03.3
kymk0yl6ba9glifm9u0lv5o22     bakumar2c.mylabserver.com   Ready    Active              19.03.3
[root@bakumar1c docker]#
```

From Node3:

```
[root@bakumar3c ~]#    docker swarm join --token
SWMTKN-1-2y2kyca449o661v9e4x7tgti4nz9bpyr5ktl1f2cnjqlpapwp6-90e9782vkg4d5xwwzesnso2r1
172.31.42.108:2377
This node joined a swarm as a worker.
```

After adding, from Manager node, list of nodes:

```
[root@bakumar1c docker]# docker node ls
ID                    HOSTNAME                STATUS    AVAILABILITY    MANAGER STATUS    ENGINE
VERSION
3b4nswbkipb25ir9f7ynbatlx *   bakumar1c.mylabserver.com   Ready    Active    Leader    19.03.3
kymk0yl6ba9glifm9u0lv5o22     bakumar2c.mylabserver.com   Ready    Active              19.03.3
t3d4asxfobykzl2jpo3c6ddui     bakumar3c.mylabserver.com   Ready    Active              19.03.3
```

If we want to add one more swarm maanger node, we can run below manager join token from any node, would be join as Manager node in swarm cluster.

```
[root@bakumar2c ~]#    docker swarm join --token
SWMTKN-1-2y2kyca449o661v9e4x7tgti4nz9bpyr5ktl1f2cnjqlpapwp6-
cmauhmmlia3hrtvbh88o9a0s8 172.31.42.108:2377
Error response from daemon: This node is already part of a swarm. Use "docker swarm
leave" to leave this swarm and join another one.
```

From Swarm Manager node, if we see info of swarm,

```
From Manager node
# docker system info | more

Server:
 Containers: 2
  Running: 2
  Paused: 0
  Stopped: 0
 Images: 2
….
…
 Swarm: active
  NodeID: 3b4nswbkipb25ir9f7ynbatlx
  Is Manager: true
  ClusterID: tny4ueeezybufj03fz6zorjwf
  Managers: 1
  Nodes: 3
….
…

From Worker Node2:
[root@bakumar2c ~]# docker system info | more
        Use `--storage-opt dm.thinpooldev` to specify a custom block storage device.
Client:
 Debug Mode: false

Server:
 Containers: 2
  Running: 1
  Paused: 0
  Stopped: 1
 Images: 1
..
..
 Swarm: active
  NodeID: kymk0yl6ba9glifm9u0lv5o22
  Is Manager: false
  Node Address: 172.31.32.77
  Manager Addresses:
   172.31.42.108:2377
 Runtimes: runc
..
..
```

Docker service create

```
[root@bakumar1c docker]# docker service

Usage:  docker service COMMAND

Manage services

Commands:
  create      Create a new service
  inspect     Display detailed information on one or more services
  logs        Fetch the logs of a service or task
  ls          List services
  ps          List the tasks of one or more services
  rm          Remove one or more services
  rollback    Revert changes to a service's configuration
  scale       Scale one or multiple replicated services
  update      Update a service

Run 'docker service COMMAND --help' for more information on a command.
[root@bakumar1c docker]# docker service ls
ID             NAME           MODE           REPLICAS       IMAGE          PORTS
<no services as of now>
```

Create a service

```
[root@bakumar1c docker]# docker service create --name backUP_App --publish 80:80 --replicas 2 httpd
43u1mkj91rex4sl1iaey0f004
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
[root@bakumar1c docker]#
```

From node2:

```
[root@bakumar2c ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAMES
79374b1036ff   httpd:latest   "httpd-foreground"   2 minutes ago   Up 2 minutes   80/tcp    backUP_App.1.v5oq157qme0oyetrvd10ds4lg
4314cc2a3d72   httpd          "httpd-foreground"   51 minutes ago   Up 51 minutes   80/tcp    testApp
[root@bakumar2c ~]# docker service ls
Error response from daemon: This node is not a swarm manager. Worker nodes can't be used to view or modify cluster state. Please run this command on a manager node or promote the current node to a manager.
```

You can run docker service commands only from docker swarm manger node.

From Manager node (Node1), you can see list of Swarm services

```
[root@bakumar1c docker]# docker service ls
ID             NAME           MODE           REPLICAS       IMAGE          PORTS
43u1mkj91rex   backUP_App     replicated     2/2            httpd:latest   *:80->80/tcp


[root@bakumar1c docker]# docker service ps backUP_App
ID             NAME           IMAGE          NODE           DESIRED STATE  CURRENT STATE  ERROR
```

Docker replica Fault tolerant:

From node2 worker:

[root@bakumar2c ~]#
[root@bakumar2c ~]# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
79374b1036ff httpd:latest "httpd-foreground" 17 minutes ago Up 17 minutes 80/tcp
backUP_App.1.v5oq157qme0oyetrvd10ds4lg
4314cc2a3d72 httpd "httpd-foreground" About an hour ago Up About an hour 80/tcp testApp
[root@bakumar2c ~]#

From Swarm node manager:

[root@bakumar1c docker]# docker service ps backUP_App
ID NAME IMAGE NODE DESIRED STATE CURRENT STATE ERROR
PORTS
v5oq157qme0o backUP_App.1 httpd:latest bakumar2c.mylabserver.com Running Running 12 seconds ago
61d0idghii01 backUP_App.2 httpd:latest bakumar1c.mylabserver.com Running Running 19 minutes ago

Stop docker from node2:

[root@bakumar2c ~]#
[root@bakumar2c ~]# systemctl stop docker
[root@bakumar2c ~]# docker ps
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?

Check it now from Node1 Swarm Manger:
(After couple of seconds)

[root@bakumar1c docker]# docker service ps backUP_App
ID NAME IMAGE NODE DESIRED STATE CURRENT STATE ERROR
PORTS
hstvwxmdkt9o backUP_App.1 httpd:latest bakumar3c.mylabserver.com **Ready** Preparing 2 seconds ago
v5oq157qme0o _ backUP_App.1 httpd:latest bakumar2c.mylabserver.com Shutdown Running 17 seconds ago
61d0idghii01 backUP_App.2 httpd:latest bakumar1c.mylabserver.com Running Running 19 minutes ago

(After few min, the failed replica will be started in another node, here node3)

[root@bakumar1c docker]# docker service ps backUP_App
ID NAME IMAGE NODE DESIRED STATE CURRENT STATE ERROR
PORTS
hstvwxmdkt9o backUP_App.1 httpd:latest **bakumar3c**.mylabserver.com **Running** Starting 2 seconds ago
v5oq157qme0o _ backUP_App.1 httpd:latest bakumar2c.mylabserver.com Shutdown Running 41 seconds ago
61d0idghii01 backUP_App.2 httpd:latest bakumar1c.mylabserver.com Running Running 20 minutes ago

From Node 3:
Initially no ps

[root@bakumar3c ~]# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

[root@bakumar3c ~]# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
50ba1310d3c5 httpd:latest "httpd-foreground" 9 seconds ago Up 4 seconds 80/tcp
backUP_App.1.hstvwxmdkt9o0v1unl9ctlawu

Docker Swarm manager -- fault-tolerant/backup

Take backup of Manager swarm directory initially as tar, then start a new node a Swarm Manager manually would bring back.

```
[root@bakumar1c docker]# cd /var/lib/docker/
[root@bakumar1c docker]# ls -lrt
total 4
drwx------. 4 root root   30 Oct 11 19:05 plugins
drwx------. 2 root root    6 Oct 11 19:05 trust
drwxr-x---. 3 root root   18 Oct 11 19:05 network
drwx------. 2 root root   23 Oct 11 19:05 builder
drwx--x--x. 4 root root   87 Oct 11 19:05 buildkit
drwx------. 2 root root    6 Oct 12 05:51 runtimes
drwx------. 2 root root   24 Oct 12 06:07 volumes
drwx------. 3 root root   25 Oct 12 06:07 image
drwx------. 5 root root   50 Oct 12 06:09 devicemapper
drwx------. 5 root root   90 Oct 12 06:45 swarm
drwx------. 5 root root 4096 Oct 12 07:50 containers
drwx------. 2 root root    6 Oct 12 07:50 tmp
[root@bakumar1c docker]# ls -lrt swarm/
total 8
-rw-------. 1 root root 213 Oct 12 06:45 docker-state.json
drwxr-xr-x. 2 root root  21 Oct 12 06:45 worker
drwxr-xr-x. 2 root root  72 Oct 12 06:45 certificates
drwx------. 4 root root  53 Oct 12 06:45 raft
-rw-------. 1 root root  69 Oct 12 06:45 state.json
```

Create a tar file for swarm directory

```
[root@bakumar1c docker]# tar -cvf swarm.tar swarm/
swarm/
swarm/docker-state.json
swarm/certificates/
swarm/certificates/swarm-root-ca.crt
swarm/certificates/swarm-node.key
swarm/certificates/swarm-node.crt
swarm/worker/
swarm/worker/tasks.db
swarm/raft/
swarm/raft/snap-v3-encrypted/
swarm/raft/wal-v3-encrypted/
swarm/raft/wal-v3-encrypted/0000000000000000-0000000000000000.wal
swarm/raft/wal-v3-encrypted/0.tmp
swarm/state.json
[root@bakumar1c docker]#
[root@bakumar1c docker]#
[root@bakumar1c docker]# ls -lrt
total 125084
drwx------. 4 root root       30 Oct 11 19:05 plugins
drwx------. 2 root root        6 Oct 11 19:05 trust
drwxr-x---. 3 root root       18 Oct 11 19:05 network
drwx------. 2 root root       23 Oct 11 19:05 builder
drwx--x--x. 4 root root       87 Oct 11 19:05 buildkit
drwx------. 2 root root        6 Oct 12 05:51 runtimes
drwx------. 2 root root       24 Oct 12 06:07 volumes
drwx------. 3 root root       25 Oct 12 06:07 image
drwx------. 5 root root       50 Oct 12 06:09 devicemapper
drwx------. 5 root root       90 Oct 12 06:45 swarm
drwx------. 5 root root     4096 Oct 12 07:50 containers
drwx------. 2 root root        6 Oct 12 07:50 tmp
-rw-r--r--. 1 root root 128081920 Oct 12 08:23 swarm.tar
[root@bakumar1c docker]# mv swarm.tar tmp/

[root@bakumar1c docker]# ls -lrt tmp/
total 125080
-rw-r--r--. 1 root root 128081920 Oct 12 08:23 swarm.tar
```

Copy this swarm tar file to newly created backup node.

Cluster sizing recommendations

Universal Control Plane (UCP) System Requirements

Breakdown of Manager Nodes for Fault Tolerance

Nodes	Accessible	Ports	Purpose
managers, workers	yes	TCP 443 (configurable)	Port for UCP's HTTPS API and web UI
managers	yes	TCP 2376 (configurable)	Port for the Docker Swarm manager. Used for backwards compatibility
managers, workers	yes	TCP 2377 (configurable)	Port for communication between swarm nodes
workers	no	TCP 2377 (configurable)	Port for communication between swarm nodes
managers, workers	no, out	UDP/TCP 4789	Port for overlay networking
managers, workers	no, out	TCP/UDP 7946	Port for gossip based clustering
managers, workers	yes	TCP 12376	Port for a TLS proxy that provides access to UCP, Docker Engine, and Docker Swarm
managers	yes	TCP 12379	Port for internal node configuration, cluster configuration, and HA
managers	yes	TCP 12380	Port for internal node configuration, cluster configuration, and HA
managers	yes	TCP 12381	Port for the cluster certificate authority
managers	yes	TCP 12382	Port for the UCP certificate authority
managers	yes	TCP 12383	Port for the authentication storage backend
managers	yes	TCP 12384	Port for the authentication storage backend for replication across managers
managers	yes	TCP 12385	Port for the authentication API
managers	yes	TCP 12386	Port for the authentication worker
managers	yes	TCP 12387	Port for the metrics service

Minimum Requirements

- 8gb RAM (Managers or DTR Nodes)
- 4gb RAM (Workers)
- 3gb Free Disk

Recommended Requirements

- 16gb RAM (Managers or DTR Nodes)
- 4vCPUs (Workers or DTR Nodes)

Requirements (UCP, DTR and Docker EE)

Performance Considerations (Timing)

Component	Timeout (ms)	Configurable
Raft consensus between manager nodes	4000	no
Gossip protocol for overlay networking	5000	no
etcd	500	yes
RethinkDB	10000	no
Vishnu alive heartbeat	30000	no

Recommendations

- Plan for Load Balancing
- Use External Certificate Authority for Production

Special Notes

- Docker EE includes
  - Docker Engine with Support
  - Docker Trusted Registry
  - Universal Control Plane
- Compatibility
  - Docker Engine 17.06+
  - DTR 2.3+
  - UCP 2.2+

```
[root@bakumar1c docker]# docker container run  --rm -it --name ucp -v /var/run/docker.sock:/var/run/docker.sock  docker/ucp:2.2.4 install --host-address 172.31.42.108 –interactive
```

Here : 172.31.42.108 is the private IP of the swarm manager node
Admin Username: admin
Admin Password: admin@12345
Confirm Admin Password: admin@12345

Uninstalling ucp from node

```
[root@bakumar1c ~]# docker container run  --rm -it --name ucp -v /var/run/docker.sock:/var/run/docker.sock  docker/ucp:2.2.4 uninstall-ucp
INFO[0007] Your engine version 19.03.3, build a872fc2f86 (3.10.0-1062.1.1.el7.x86_64) is compatible
INFO[0009] We detected UCP instance tny4ueeezybufj03fz6zorjwf
FATA[0009] re-run the command with "--id tny4ueeezybufj03fz6zorjwf" or --interactive to confirm you want to remove this UCP instance
[root@bakumar1c ~]#
[root@bakumar1c ~]#
[root@bakumar1c ~]# docker container run  --rm -it --name ucp -v /var/run/docker.sock:/var/run/docker.sock  docker/ucp:2.2.4 uninstall-ucp  --interactive
INFO[0000] Your engine version 19.03.3, build a872fc2f86 (3.10.0-1062.1.1.el7.x86_64) is compatible
INFO[0000] We're about to uninstall from this swarm cluster. UCP ID: tny4ueeezybufj03fz6zorjwf
Do you want to proceed with the uninstall? (y/n):

FATA[0008] not uninstalling per user request
[root@bakumar1c ~]#
[root@bakumar1c ~]#
[root@bakumar1c ~]# docker container run  --rm -it --name ucp -v /var/run/docker.sock:/var/run/docker.sock  docker/ucp:2.2.4 uninstall-ucp  --interactive
INFO[0000] Your engine version 19.03.3, build a872fc2f86 (3.10.0-1062.1.1.el7.x86_64) is compatible
INFO[0001] We're about to uninstall from this swarm cluster. UCP ID: tny4ueeezybufj03fz6zorjwf
Do you want to proceed with the uninstall? (y/n): y
INFO[0025] Uninstalling UCP on each node...
INFO[0122] UCP has been removed from this cluster successfully.
INFO[0124] Removing UCP Services
[root@bakumar1c ~]#
```

If installation takes longer time, and fails with below errrors:

```
[root@bakumar1c ~]# docker container run  --rm -it --name ucp -v /var/run/docker.sock:/var/run/docker.sock  docker/ucp:2.2.4 uninstall-ucp  --interactive
INFO[0000] Your engine version 19.03.3, build a872fc2f86 (3.10.0-1062.1.1.el7.x86_64) is compatible
INFO[0000] We're about to uninstall from this swarm cluster. UCP ID: tny4ueeezybufj03fz6zorjwf
Do you want to proceed with the uninstall? (y/n): y
INFO[0001] Uninstalling UCP on each node...
ERRO[0144] Uninstalling UCP took too long!
ERRO[0144] The following nodes have failed to uninstall within the timeout: 3b4nswbkipb25ir9f7ynbatlx
ERRO[0144] Connect to these nodes and ensure that there are no UCP containers or volumes present.
ERRO[0144] Then remove the `ucp-agent`, `ucp-agent-s390x`, and `ucp-agent-win` services from a swarm manager.
FATA[0144] failed to cleanly uninstall UCP
```

Then uninstall UCP manually:

Refernces:

https://docs.docker.com/v17.09/datacenter/ucp/1.1/installation/uninstall/#example

https://docs.docker.com/ee/ucp/admin/install/uninstall/

```
docker service rm ucp-agent-win ucp-agent-s390x ucp-agent
docker ps -a | grep ucp | awk '{print $1}' | xargs docker rm -f
docker volume ls | grep ucp | awk '{print $2}' | xargs docker volume rm
```

[root@bakumar1c ~]# docker rmi docker/ucp
Untagged: docker/ucp:latest
Untagged: docker/ucp@sha256:5732446035636c07f1b5f133827fc464da61990cf37c9ef850cff9012ec2bb0d
Deleted: sha256:131ec53900dbf25b9110edd2da0eefca9c3ee09c088ee7c7d4d3010cff155c7d
Deleted: sha256:7102dad44ff349a7121948a3e87aa8f38e065b0d56ec7d2fdb7c80dbed421fc5
Deleted: sha256:b4e6ccfc8585d57faef217d8ee29ed974fb8ec41b5106642624362f8a5c4cf56
Deleted: sha256:1bfeebd65323b8ddf5bd6a51cc7097b72788bc982e9ab3280d53d3c613adffa7
[root@bakumar1c ~]#

[root@bakumar1c ~]# docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	f949e7d76d63	2 weeks ago	126MB
httpd	latest	19459a872194	4 weeks ago	154MB
centos	6	d0957ffdf8a2	7 months ago	194MB
docker/ucp-auth	2.2.4	40e5725374ac	23 months ago	20.2MB
docker/ucp-controller	2.2.4	0dd8233eed53	23 months ago	38.6MB
docker/ucp-metrics	2.2.4	57797febc06b	23 months ago	79.6MB
docker/ucp-hrm	2.2.4	b7b61db92308	23 months ago	7.12MB
docker/ucp-etcd	2.2.4	d51f71163016	23 months ago	29.6MB
docker/ucp-dsinfo	2.2.4	3172cba25055	23 months ago	548MB
docker/ucp-compose	2.2.4	ed9aee006cf0	23 months ago	39.6MB
docker/ucp-auth-store	2.2.4	ea0bf03a897d	23 months ago	60.5MB
docker/ucp-cfssl	2.2.4	ebdf8c052738	23 months ago	12.1MB
docker/ucp-agent	2.2.4	f149286db4e6	23 months ago	20.1MB
docker/ucp	2.2.4	133ae4e2494d	23 months ago	19.9MB
docker/ucp-swarm	2.2.4	8f4ee7209ae0	23 months ago	21.6MB

[root@bakumar1c ~]# docker rmi docker/ucp
Error: No such image: docker/ucp

[root@bakumar1c ~]# docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
428fce517987	docker/ucp-agent:2.2.4	"/bin/ucp-agent agent"	14 seconds ago	Up 8 seconds	2376/tcp	ucp-agent.3b4nswbkipb25ir9f7ynbatlx.qd6smvipfd3m5hifhlokq80h7

[root@bakumar1c ~]# docker service rm ucp-agent-win ucp-agent-s390x ucp-agent
ucp-agent-win
ucp-agent-s390x
ucp-agent

[root@bakumar1c ~]# docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES

[root@bakumar1c ~]# docker ps -a

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
fe12c2938fb7	httpd:latest	"httpd-foreground"	2 hours ago	Exited (0) 49 minutes ago		backUP_App.2.w3219zrf9qu54k3x9zkmxddgs
c84b5cc3f015	httpd:latest	"httpd-foreground"	2 hours ago	Dead		backUP_App.2.tnah23sslv669bp53tnkghcpp
a74d74428e39	httpd	"httpd-foreground"	8 hours ago	Exited (0) 2 hours ago		testSyslog
113d82c4ec74	httpd	"httpd-foreground"	8 hours ago	Exited (0) 2 hours ago		testApp

Finally ucp installation is success

```
[root@bakumar1c ~]# docker container run  --rm -it --name ucp -v /var/run/docker.sock:/var/run/docker.sock  docker/ucp:2.2.4 install --host-address 172.31.42.108 –-interactive
INFO[0000] Verifying your system is compatible with UCP 2.2.4 (168ec746e)
INFO[0000] Your engine version 19.03.3, build a872fc2f86 (3.10.0-1062.1.1.el7.x86_64) is compatible
WARN[0000] Your system does not have enough memory.  UCP suggests a minimum of 2.00 GB, but you only have 0.99 GB.  You may have unexpected errors.
WARN[0002] Your system uses devicemapper.  We can not accurately detect available storage space.  Please make sure you have at least 3.00 GB available in /var/lib/docker
INFO[0004] All required images are present
INFO[0000] Found existing UCP config com.docker.ucp.config-4
WARN[0000] UCP will continue the install with config com.docker.ucp.config-4 in 10 seconds. Press Ctrl-C to cancel this installation. You can skip this pause with the --existing-config flag.
INFO[0024] Establishing mutual Cluster Root CA with Swarm
INFO[0024] Installing UCP with host address 172.31.42.108 - If this is incorrect, please specify an alternative address with the '--host-address' flag
INFO[0024] Generating UCP Client Root CA
INFO[0025] Deploying UCP Service
INFO[0127] Installation completed on bakumar1c.mylabserver.com (node 3b4nswbkipb25ir9f7ynbatlx)
INFO[0134] Installation completed on bakumar2c.mylabserver.com (node kymk0yl6ba9glifm9u0lv5o22)
INFO[0149] Installation completed on bakumar3c.mylabserver.com (node t3d4asxfobykzl2jpo3c6ddui)
INFO[0149] UCP Instance ID: tny4ueeezybufj03fz6zorjwf
INFO[0149] UCP Server SSL: SHA-256
Fingerprint=9A:AC:A6:26:1E:7B:B1:E3:C5:19:69:12:DC:67:F2:43:6D:AC:CC:AB:80:FC:41:5C:16:44:44:25:99:03:8D:C3
INFO[0149] Login to UCP at https://172.31.42.108:443
INFO[0149] Username: admin
INFO[0149] Password: nUcIRLRg5TIRyUYNWhKMt9ZcbzRz+hB5
```

As earlier seen , logging into ucp site using :
https://ucp.example.com
provide uname and pwd as mentioned in above output:

```
[root@bakumar1c ~]# docker ps
CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS              PORTS
NAMES
763703c04c16        docker/ucp-agent:2.2.4        "/bin/ucp-agent agent"   4 minutes ago       Up 4 minutes            2376/tcp
ucp-agent.3b4nswbkipb25ir9f7ynbatlx.cz5plfkjkavk5ajozy247rkz3
6a45cfcb0725        docker/ucp-controller:2.2.4   "/bin/controller ser…"   5 minutes ago       Up 5 minutes (healthy)   0.0.0.0:443->8080/tcp
ucp-controller
fca953b8b1f5        docker/ucp-auth:2.2.4         "/usr/local/bin/enzi…"   5 minutes ago       Up 5 minutes (healthy)   0.0.0.0:12385->4443/tcp
ucp-auth-api
07e3c5da3cbf        docker/ucp-swarm:2.2.4        "/bin/swarm manage -…"   5 minutes ago       Up 5 minutes (healthy)   0.0.0.0:2376->2375/tcp
ucp-swarm-manager
f2dcc48ea923        docker/ucp-metrics:2.2.4      "/bin/entrypoint.sh …"   5 minutes ago       Up 5 minutes (healthy)   0.0.0.0:12387->12387/tcp
ucp-metrics
38a8b17b16f5        docker/ucp-auth:2.2.4         "/usr/local/bin/enzi…"   5 minutes ago       Up 5 minutes (healthy)   0.0.0.0:12386->4443/tcp
ucp-auth-worker
d79571690fb5        docker/ucp-auth-store:2.2.4   "rethinkdb --bind al…"   5 minutes ago       Up 5 minutes (healthy)   0.0.0.0:12383-12384->12383-12384/tcp
ucp-auth-store
f058daf47434        docker/ucp-etcd:2.2.4         "/bin/etcd --data-di…"   6 minutes ago       Up 6 minutes (healthy)   2380/tcp, 4001/tcp, 7001/tcp, 0.0.0.0:12380->12380/
tcp, 0.0.0.0:12379->2379/tcp   ucp-kv
e67c2fc06320        docker/ucp-cfssl:2.2.4        "/bin/ucp-ca serve -…"   6 minutes ago       Up 6 minutes (healthy)   0.0.0.0:12381->12381/tcp
ucp-cluster-root-ca
f771d4adc096        docker/ucp-cfssl:2.2.4        "/bin/ucp-ca serve -…"   6 minutes ago       Up 6 minutes (healthy)   0.0.0.0:12382->12382/tcp
ucp-client-root-ca
ec10ec0d72bf        docker/ucp-agent:2.2.4        "/bin/ucp-agent prox…"   6 minutes ago       Up 6 minutes (healthy)   0.0.0.0:12376->2376/tcp
ucp-proxy
2f81af9b3b7c        httpd:latest                  "httpd-foreground"       20 minutes ago      Up 20 minutes           80/tcp
backUP_App.2.dncqv7pmrxo0m61ojxdvo9j8t
[root@bakumar1c ~]#
```

Docker NameSpaces & Control groups(cgroups)

Docker and Namespaces

Namespaces provide 'isolation' so that other pieces of the system remain unaffected by whatever is within that namespace.

Docker uses namespaces of various kinds to provide the isolation that containers need in order to remain portable and refrain from affecting the remainder of the host system.

Namespace Types

- Process ID
- Mount
- IPC
- User (currently experimental support for)
- Network

Linux Academy | CloudAssessments

Docker and Control Groups (cgroups)

Control Groups provide resource limitation and reporting capability within the container space. They allow granular control over what host resources are allocated to the container(s) and when they are allocated.

Common Control Groups

- CPU
- Memory
- Network Bandwidth
- Disk
- Priority

Linux Academy | CloudAssessments

Docker Images

```
[root@bakumar2c ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker/dtr          latest              127043314e23        3 days ago          75.2MB
httpd               latest              19459a872194        4 weeks ago         154MB
hello-world         latest              fce289e99eb9        9 months ago        1.84kB
hello-world         linux               fce289e99eb9        9 months ago        1.84kB
docker/ucp-agent    2.2.4               f149286db4e6        23 months ago       20.1MB
```

```
[root@bakumar2c ~]# docker images --digests
REPOSITORY          TAG                 DIGEST                                                                    IMAGE ID            CREATED             SIZE
docker/dtr          latest              sha256:e1eae7579a6a1793d653dd97df9297464600e2644565fa2ed33c351f942facf6   127043314e23        3 days ago          75.2MB
httpd               latest              sha256:39d7d9a3ab93c0ad68ee7ea237722ed1b0016ff6974d80581022a53ec1e58797   19459a872194        4 weeks ago         154MB
hello-world         latest              sha256:31ab99b55a32ab6dd0dea8044dc26f89e3c854bf07e47e021bdb69f5c8a918b3   fce289e99eb9        9 months ago        1.84kB
hello-world         latest              sha256:c3b4ada4687bbaa170745b3e4dd8ac3f194ca95b2d0518b417fb47e5879d9b5f   fce289e99eb9        9 months ago        1.84kB
hello-world         linux               sha256:31ab99b55a32ab6dd0dea8044dc26f89e3c854bf07e47e021bdb69f5c8a918b3   fce289e99eb9        9 months ago        1.84kB
hello-world         linux               sha256:c3b4ada4687bbaa170745b3e4dd8ac3f194ca95b2d0518b417fb47e5879d9b5f   fce289e99eb9        9 months ago        1.84kB
docker/ucp-agent    2.2.4               sha256:bbee2b38e355e613c6b740698fba887d65f77b3c69e81bd9eebd4ae6665e43c5   f149286db4e6        23 months ago       20.1MB
```

only image names

```
[root@bakumar2c ~]# docker images -q
127043314e23
19459a872194
fce289e99eb9
fce289e99eb9
f149286db4e6
```

Image names with full name without truncating

```
[root@bakumar2c ~]# docker images --no-trunc
REPOSITORY          TAG                 IMAGE ID                                                                  CREATED             SIZE
docker/dtr          latest              sha256:127043314e239d5e80e4e3fb809855b56c993288568fbb3cf18c00bc744d3a20   4 days ago          75.2MB
httpd               latest              sha256:19459a87219415cc5751fb48e9aa37ea580b98eed1fe784e76c4bc3c9b3b0741   4 weeks ago         154MB
hello-world         latest              sha256:fce289e99eb9bca977dae136fbe2a82b6b7d4c372474c9235adc1741675f587e   9 months ago        1.84kB
hello-world         linux               sha256:fce289e99eb9bca977dae136fbe2a82b6b7d4c372474c9235adc1741675f587e   9 months ago        1.84kB
docker/ucp-agent    2.2.4               sha256:f149286db4e66d0f5dbeb0d700cb8098c168383b1863a273289ff30df26aff3f   23 months ago       20.1MB
[root@bakumar2c ~]#
```

Images filters

```
[root@bakumar2c ~]#
[root@bakumar2c ~]# docker images --filter "before=hello-world"
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker/ucp-agent    2.2.4               f149286db4e6        23 months ago       20.1MB
```

```
[root@bakumar2c ~]# docker images --filter "before=httpd"
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              fce289e99eb9        9 months ago        1.84kB
hello-world         linux               fce289e99eb9        9 months ago        1.84kB
docker/ucp-agent    2.2.4               f149286db4e6        23 months ago       20.1MB
```

```
[user@tcox1 ~]$ docker images --no-trunc
REPOSITORY          TAG                 IMAGE ID                                                                  CREAT
ED          SIZE
centos              6                   sha256:d3704b1d21541048 9ca2bf883597 8e 0054546a71c 39309c910 7440ac02d527f   9 day
s ago       194MB
hello-world         latest              sha256:f2a91732366c0332ccd7afd2a6c4ff2b9af81f5413706f7a19acd460f87685bc7   2 wee
ks ago      1.85kB
hello-world         linux               sha256:f2a91732366c0332ccd7afd2a6c4ff2b9af81f5413706f7a19acd460f87685bc7   2 wee
ks ago      1.85kB
[user@tcox1 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              6                   d3704b1d215         9 days ago          194MB
hello-world         latest              f2a91732366c        2 weeks ago         1.85kB
hello-world         linux               f2a91732366c        2 weeks ago         1.85kB
[user@tcox1 ~]$
```

```
# This is our first real Dockerfile
FROM centos:6

LABEL maintainer="latest@linuxacademy.com"

RUN yum update -y && yum install http net tools
RUN mkdir -p /run/http
RUN rm -rf /run/http/* /tmp/httpd*

CMD echo "Remember to check your container IP Address"

-- INSERT --                                          12,1          Bot
```

Dockerfile - Options and Structure

ENV
<key> <value>
<key>=<value>
ADD
<src> ... <dest> (must include quotes if spaces)
WORKDIR - used as relative for relative path
COPY
<src> ... <dest> (must include quotes if spaces)
WORKDIR - used as relative for relative path
ENTRYPOINT
["command", "param1", "param2"] (exec form, JSON)
command param1 param2 (shell form)
VOLUME
["/var/"]
USER <user>[:<group>] or USER [<UID>[:<GID>]]
affects RUN, CMD & ENTRYPOINT
WORKDIR
/path/to/dir
setsworking directory for RUN, CMD, ENTRYPOINT

```
[user@xxxx Dockerfiles]$ docker build -t mywebserver:1 .
Sending build context to Docker daemon  2.048kB
Step 1/9 : FROM centos:5
 ---> d7328fa6abb9
Step 2/9 : LABEL maintainer "latest@linuxacademy.com"
 ---> Using cache
 ---> 1e2e90bf45a3
Step 3/9 : RUN yum update -y && yum install -y httpd net-tools
 ---> Running in 1f236f3c676
Loaded plugins: fastestmirror, ovl
Setting up Update Process
```

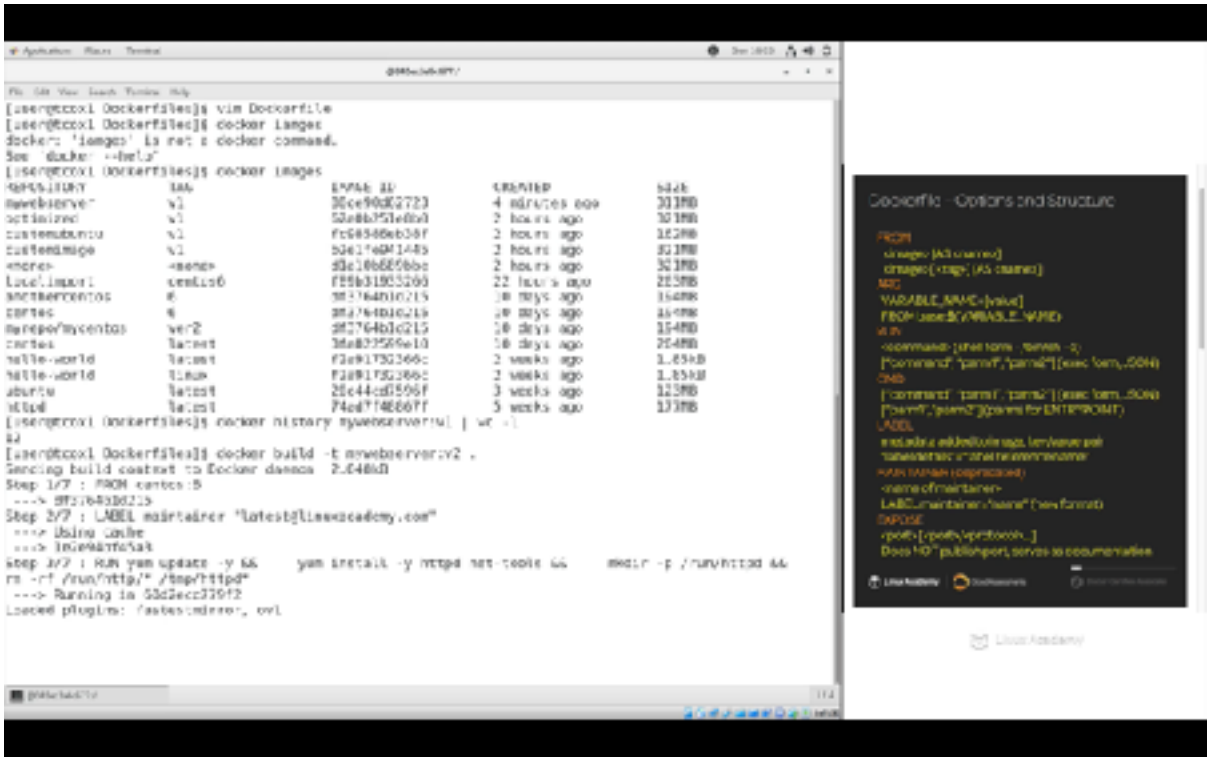
Dockerfile - Options and Structure

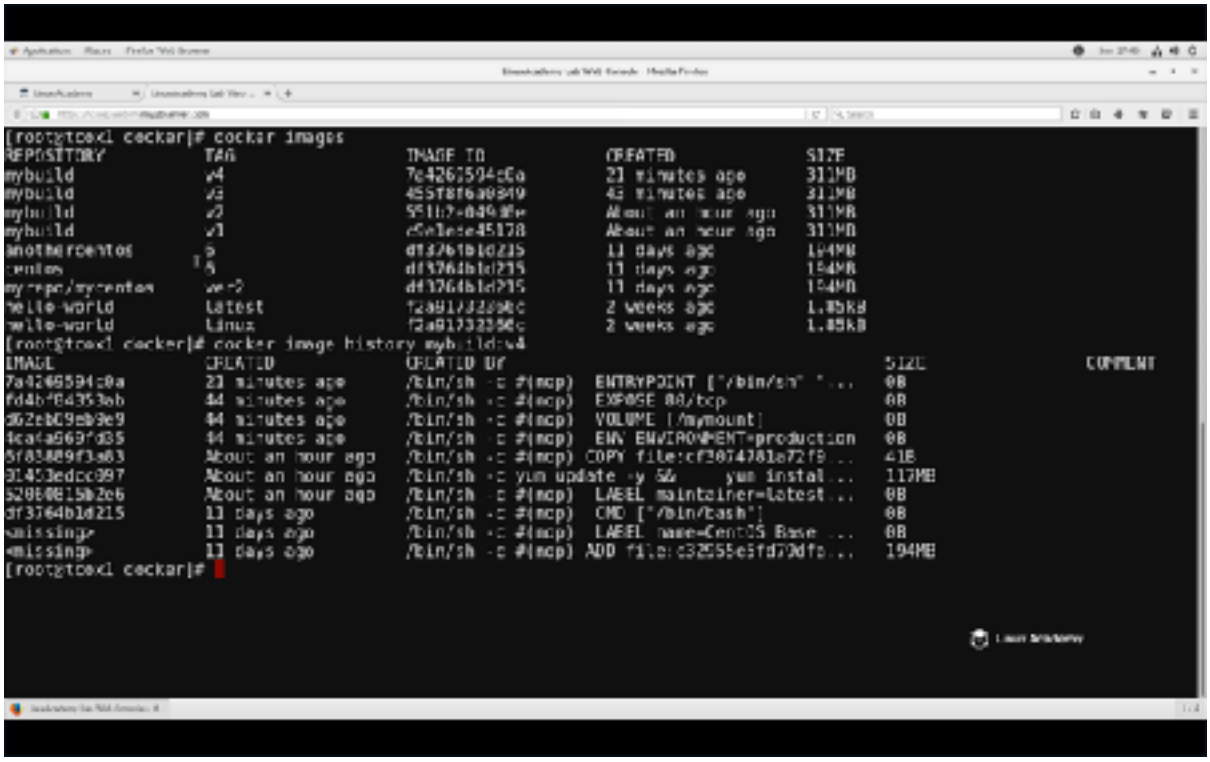
```
FROM
  <image> [AS <name>]
  <image>[:<tag>] [AS <name>]
ARG
  VARIABLE_NAME=[value]
  FROM <base${VARIABLE_NAME}>
RUN
  <command> (shell form, runs in a
  ["command", "param1", "param2"] (exec form, JSON)
CMD
  ["command", "param1", "param2"] (exec form, JSON)
  ["param1", "param2"] (params for ENTRYPOINT)
ADD
  <source> <dest> [<chown>] [<chmod>] [<checksum>]
  <checksum> = sha256:<hexadecimal_hash>
MAINTAINER (deprecated)
  <name of maintainer>
  LABEL maintainer="<name>" (new format)
EXPOSE
  <port> [<port>/<protocol>...]
  Does NOT publish port, serves as documentation
```

```
# This is our first real Dockerfile
FROM centos:5

LABEL maintainer="latest@linuxacademy.com"

RUN yum update -y && \
    yum install -y httpd net-tools && \
    mkdir -p /var/run/httpd && \
    rm -rf /run/http/* /tmp/httpd*

CMD echo "Remember to check your container IP Address"

ENV ENVIRONMENT="production"

EXPOSE 80

ENTRYPOINT apachectl "-DFOREGROUND"
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                    7,5           All
```

Dockerfile - Options and Structure

```
FROM
  <image> [AS <name>]
  <image>[:<tag>] [AS <name>]
ARG
  VARIABLE_NAME=[value]
  FROM <base${VARIABLE_NAME}>
RUN
  <command> (shell form, runs in a
  ["command", "param1", "param2"] (exec form, JSON)
CMD
  ["command", "param1", "param2"] (exec form, JSON)
  ["param1", "param2"] (params for ENTRYPOINT)
ADD
  <source> <dest> [<chown>] [<chmod>] [<checksum>]
  <checksum> = sha256:<hexadecimal_hash>
MAINTAINER (deprecated)
  <name of maintainer>
  LABEL maintainer="<name>" (new format)
EXPOSE
  <port> [<port>/<protocol>...]
  Does NOT publish port, serves as documentation
```


[user@tcccl Dockerfiles]$ vim Dockerfile
[user@tcccl Dockerfiles]$ docker images
docker: 'images' is not a docker command.
See 'docker --help'
[user@tcccl Dockerfiles]$ docker images
[user@tcccl Dockerfiles]$ docker build -t mywebserver:v2 .
Sending build context to Docker daemon 2.048kB
Step 1/7 : FROM centos:5
Step 2/7 : LABEL maintainer "latest@linuxacademy.com"
Step 3/7 : RUN yum update -y && yum install -y httpd net-tools && mkdir -p /var/run/httpd &&
rm -rf /run/*/* /tmp/*/*
Loaded plugins: fastestmirror, ovl
Dockerfile - Options and Structure
FROM
VARIABLE_NAME=[value]
ENV
CMD
LABEL
EXPOSE


[root@tcccl docker]# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
mybuild v4 7e4265594c0a 21 minutes ago 311MB
mybuild v3 455f8f6a0849 43 minutes ago 311MB
mybuild v2 5511b7e049fe About an hour ago 311MB
mybuild v1 e5e1ece45178 About an hour ago 311MB
anothercentos 5 df3764b1d215 11 days ago 194MB
centos 5 df3764b1d215 11 days ago 194MB
myrepo/mycentos ver2 df3764b1d215 11 days ago 194MB
hello-world latest f2a91732366c 2 weeks ago 1.85kB
hello-world Linux f2a91732366c 2 weeks ago 1.85kB
[root@tcccl docker]# docker image history mybuild:v4
IMAGE CREATED CREATED BY SIZE COMMENT
7e4265594c0a 21 minutes ago /bin/sh -c #(nop) ENTRYPOINT ["/bin/sh" "... 0B
fd4b7843536b 44 minutes ago /bin/sh -c #(nop) EXPOSE 80/tcp 0B
362ebb9b9e9 44 minutes ago /bin/sh -c #(nop) VOLUME [/mymount] 0B
4ca4a669fd35 44 minutes ago /bin/sh -c #(nop) ENV ENVIRONMENT=production 0B
5f83889f3a83 About an hour ago /bin/sh -c #(nop) COPY file:cf3074781a72f9 ... 41B
31431edcc097 About an hour ago /bin/sh -c yum update -y && yum instal... 117MB
52080815b2e6 About an hour ago /bin/sh -c #(nop) LABEL maintainer=latest... 0B
df3764b1d215 11 days ago /bin/sh -c #(nop) CMD ["/bin/bash"] 0B
<missing> 11 days ago /bin/sh -c #(nop) LABEL name=CentOS Base ... 0B
<missing> 11 days ago /bin/sh -c #(nop) ADD file:c32555e5fd70dfb... 194MB
[root@tcccl docker]#

```
[root@tcxl ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mybuild             importv5            a4e3a1a2ef30        16 seconds ago      268MB
mybuild             v4                  7a426b594c6a        About an hour ago   311MB
mybuild             v3                  455f8f6a6949        About an hour ago   311MB
mybuild             v2                  551b7e0494de        2 hours ago         311MB
mybuild             v1                  c5e1e1e45178        2 hours ago         311MB
another-centos      5                   d13764b1d715        11 days ago         194MB
centos              5                   d13764b1d715        11 days ago         194MB
myrepo/mycentos     v0.2                d13764b1d715        11 days ago         194MB
hello-world         latest              f2a91732366c        2 weeks ago         1.85kB
hello-world         linux               f2a91732366c        2 weeks ago         1.85kB
[root@tcxl ~]# docker image history mybuild:importv5
IMAGE               CREATED             CREATED BY          SIZE                COMMENT
a4e3a1a2ef30        30 seconds ago                          268MB               Imported from -
[root@tcxl ~]# docker image history mybuild:v4
IMAGE               CREATED             CREATED BY                                      SIZE                COMMENT
7a426b594c6a        About an hour ago   /bin/sh -c #(nop)  ENTRYPOINT ["/bin/sh" "...   0B
fd4b1b4353ab        About an hour ago   /bin/sh -c #(nop)  EXPOSE 80/tcp                0B
d62cb69cb9e9        About an hour ago   /bin/sh -c #(nop)  VOLUME [/mymount]            0B
4ca4a969fd35        About an hour ago   /bin/sh -c #(nop)  ENV ENVIRONMENT=production   0B
5f838b9f3a83        2 hours ago         /bin/sh -c #(nop) COPY file:cf3074781a72f9...   41B
31453cdcc997        2 hours ago         /bin/sh -c yum update -y &&     yum instal...   117MB
520608156206        2 hours ago         /bin/sh -c #(nop)  LABEL maintainer=latest ...  0B
df3764b1d715        11 days ago         /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>           11 days ago         /bin/sh -c #(nop)  LABEL name=CentOS Base ...   0B
<missing>           11 days ago         /bin/sh -c #(nop) ADD file:c32655e5fd70dfa...   194MB
[root@tcxl ~]#
```

```
[user@tcxl docker]$ docker info | grep Storage
WARNING: overlay: the backing xfs filesystem is formatted without d_type support, which leads to incorrect behavior.
         Reformat the filesystem with ftype=1 to enable d_type support.
         Running without d_type support will not be supported in future releases.
Storage Driver: overlay
[user@tcxl docker]$
```

```
[root@tcxl ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              5                   d3764b1d235         11 days ago         124MB
[root@tcxl ~]# docker info | grep Storage
WARNING: devicemapper: usage of loopback devices is strongly discouraged for production use.
         Use `--storage-opt dm.thinpooldev` to specify a custom block storage device.
Storage Driver: devicemapper
[root@tcxl ~]#
```

```
[user@tcxl ~]$ mkdir certs
[user@tcxl ~]$ mkdir auth
[user@tcxl ~]$ openssl req -newkey rsa:4096 -nodes -sha256 -keyout certs/dockerrepo.key -x509 -days 365 -out certs/doc
kerrepo.crt -subj /CN=myregistrydomain.com
Generating a 4096 bit RSA private key
.++++++
```

```
[user@tcex1 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              5                   d13764b1d215        12 days ago         104MB
[user@tcex1 ~]$ docker pull registry:2
2: Pulling from library/registry
b7e51e37a18: Pull complete
e8ad8113ce25: Pull complete
58084b5bce2f: Pull complete
f6000c7b276c: Pull complete
f792fccd8ff6: Pull complete
Digest: sha256:9d295869d138b0b2552f9c78c9158878af5c9a9c15a3fbd1351df03eaad54c6a
Status: Downloaded newer image for registry:2
[user@tcex1 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
registry            2                   177991bc f8b2       11 days ago         33.9MB
centos              5                   d13764b1d215        12 days ago         104MB
[user@tcex1 ~]$
```

```
[user@tcex5 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nttpd               latest              723061520545        3 weeks ago         177MB
[user@tcex5 ~]$ docker run -d --name testweb nttpd
bc419f2b5793ef8c74b631613ce444c8ed6ece14e285c2112fe7e4fcbf8a2a2
[user@tcex5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
  NAMES
bc419f2b5793        nttpd               "nttpd -foreground"   5 seconds ago       Up 4 seconds        80/tcp
  testweb
[user@tcex5 ~]$ docker stop testweb
testweb
[user@tcex5 ~]$ docker start testweb
testweb
[user@tcex5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
  NAMES
bc419f2b5793        nttpd               "nttpd -foreground"   21 seconds ago      Up 1 second         80/tcp
  testweb
[user@tcex5 ~]$
```

```
[user@tcox5 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
httpd               latest              7230615c0545        3 weeks ago         177MB
[user@tcox5 ~]$ docker run -d --name testweb httpd
bc419f2b5793ef8c74b631612ce444c8e8cecec14e285c2172e7e4febf8a2a2
[user@tcox5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
  NAMES
bc419f2b5793        httpd               "httpd-foreground"  5 seconds ago       Up 4 seconds        80/tcp
  testweb
[user@tcox5 ~]$ docker stop testweb
testweb
[user@tcox5 ~]$ docker start testweb
testweb
[user@tcox5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
  NAMES
bc419f2b5793        httpd               "httpd-foreground"  21 seconds ago      Up 1 second         80/tcp
  testweb
[user@tcox5 ~]$ sudo systemctl restart docker
[user@tcox5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
 NAMES
[user@tcox5 ~]$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS
        NAMES
bc419f2b5793        httpd               "httpd-foreground"  42 seconds ago      Exited (0) 8 seconds ago
        testweb
[user@tcox5 ~]$
```

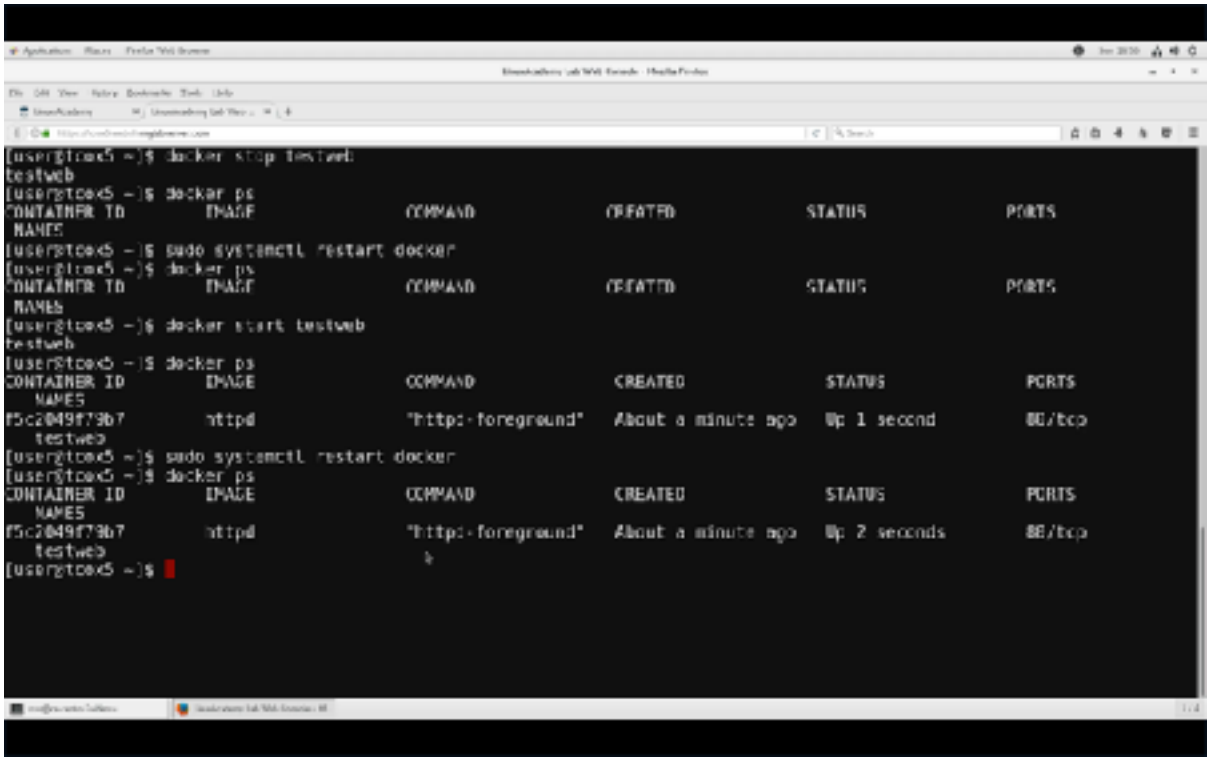
```
[user@tcox5 ~]$ docker container run -d --name testweb --restart always httpd
5429a84a2d403a6d5f55e3c6e27db0e9540f23938627f5c8e3454ec354314405
[user@tcox5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
  NAMES
5429a84a2d40        httpd               "httpd-foreground"  4 seconds ago       Up 2 seconds        80/tcp
  testweb
[user@tcox5 ~]$ sudo systemctl restart docker
[user@tcox5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
  NAMES
5429a84a2d40        httpd               "httpd-foreground"  22 seconds ago      Up 7 seconds        80/tcp
  testweb
[user@tcox5 ~]$
```

```
[user@tcox5 ~]$ docker container run -d --name testweb --restart always httpd
5429a84a2d40a6d5f55e3c6e77b7e9840f23938627f5c8e3454ec354314405
[user@tcox5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND              CREATED             STATUS              PORTS
  NAMES
5429a84a2d40        httpd               "httpd-foreground"   4 seconds ago       Up 2 seconds        80/tcp
  testweb
[user@tcox5 ~]$ sudo systemctl restart docker
[user@tcox5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND              CREATED             STATUS              PORTS
  NAMES
5429a84a2d40        httpd               "httpd-foreground"   22 seconds ago      Up 7 seconds        80/tcp
  testweb
[user@tcox5 ~]$ docker stop testweb
testweb
[user@tcox5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND              CREATED             STATUS              PORTS
 NAMES
[user@tcox5 ~]$
```

```
[user@tcox5 ~]$ docker container run -d --name testweb --restart always httpd
5429a84a2d40a6d5f55e3c6e77b7e9840f23938627f5c8e3454ec354314405
[user@tcox5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND              CREATED             STATUS              PORTS
  NAMES
5429a84a2d40        httpd               "httpd-foreground"   4 seconds ago       Up 2 seconds        80/tcp
  testweb
[user@tcox5 ~]$ sudo systemctl restart docker
[user@tcox5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND              CREATED             STATUS              PORTS
  NAMES
5429a84a2d40        httpd               "httpd-foreground"   22 seconds ago      Up 7 seconds        80/tcp
  testweb
[user@tcox5 ~]$ docker stop testweb
testweb
[user@tcox5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND              CREATED             STATUS              PORTS
 NAMES
[user@tcox5 ~]$ sudo systemctl restart docker
[user@tcox5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND              CREATED             STATUS              PORTS
  NAMES
5429a84a2d40        httpd               "httpd-foreground"   About a minute ago  Up 2 seconds        80/tcp
  testweb
[user@tcox5 ~]$
```

```
[user@tcex5 ~]$ docker container run -d --name testweb --restart always httpd
5429a84a2d40a6d5f55c3c6e77b7e9840f23038627f3c8e3454ec354314405
[user@tcex5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND              CREATED             STATUS              PORTS
  NAMES
5429a84a2d40        httpd               "httpd-foreground"   4 seconds ago       Up 2 seconds        80/tcp
  testweb
[user@tcex5 ~]$ sudo systemctl restart docker
[user@tcex5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND              CREATED             STATUS              PORTS
  NAMES
5429a84a2d40        httpd               "httpd-foreground"   22 seconds ago      Up 7 seconds        80/tcp
  testweb
[user@tcex5 ~]$ docker stop testweb
testweb
[user@tcex5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND              CREATED             STATUS              PORTS
 NAMES
[user@tcex5 ~]$ sudo systemctl restart docker
[user@tcex5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND              CREATED             STATUS              PORTS
   NAMES
5429a84a2d40        httpd               "httpd-foreground"   About a minute ago  Up 2 seconds        80/tcp
   testweb
[user@tcex5 ~]$ docker stop testweb
testweb
[user@tcex5 ~]$ docker rm testweb
testweb
[user@tcex5 ~]$ clea
```

```
[user@tcex5 ~]$ docker container run -d --name testweb httpd
9cde478d47d71c1fc472e7dd1c82e2b2c9bc4f8357lB8ia02e5c82Ca46db05b0
[user@tcex5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND              CREATED             STATUS              PORTS
  NAMES
9cde478d47d7        httpd               "httpd-foreground"   3 seconds ago       Up 1 second         80/tcp
  testweb
[user@tcex5 ~]$ docker stop testweb
testweb
[user@tcex5 ~]$ docker rm testweb
testweb
[user@tcex5 ~]$ docker container run -d --name testweb --restart unless-stopped httpd
f5c2049f79b7f4dd32d1fc92fdb54ba1af33e5a9f987c5ia193246e20e7767ac7
[user@tcex5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND              CREATED             STATUS              PORTS
  NAMES
f5c2049f79b7        httpd               "httpd-foreground"   4 seconds ago       Up 3 seconds        80/tcp
  testweb
[user@tcex5 ~]$ sudo systemctl restart docker
[user@tcex5 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND              CREATED             STATUS              PORTS
  NAMES
f5c2049f79b7        httpd               "httpd-foreground"   20 seconds ago      Up 2 seconds        80/tcp
  testweb
[user@tcex5 ~]$ e
```

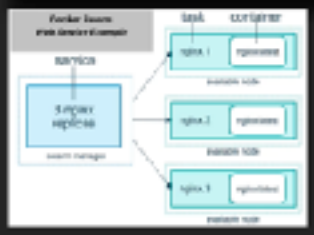


Container and Service:

Services – docker service

Using 'docker service' is the solution to managing containers deployed in highly available, easily scalable cluster implementations.

We can now think of a service as something that is consumed within the context of a larger application (which can include other Docker Services in its makeup).



Whereas containers are limited to the single host they are started on; services are containers that live on a scalable number of 'workers' in a cluster of systems. Docker Swarm handles access to, and the availability of, that service across these worker nodes, eliminating the challenges of routing and accessing individual containers.

Scalability is key in the enterprise, both up and down, in order to maximize your infrastructure spend. The scalability services allow you to take granular control of your CPU, Memory, Disk, Network, and more.

Quick Hits

Containers

- Encapsulate an application or function.
- Run on a single host at a time.
- Require manual steps to expose functionality outside of the host system (ports, network and/or volumes).
- Require more complex configuration to use multiple instances (proxies for example).
- Not highly available.
- Not easily scalable.

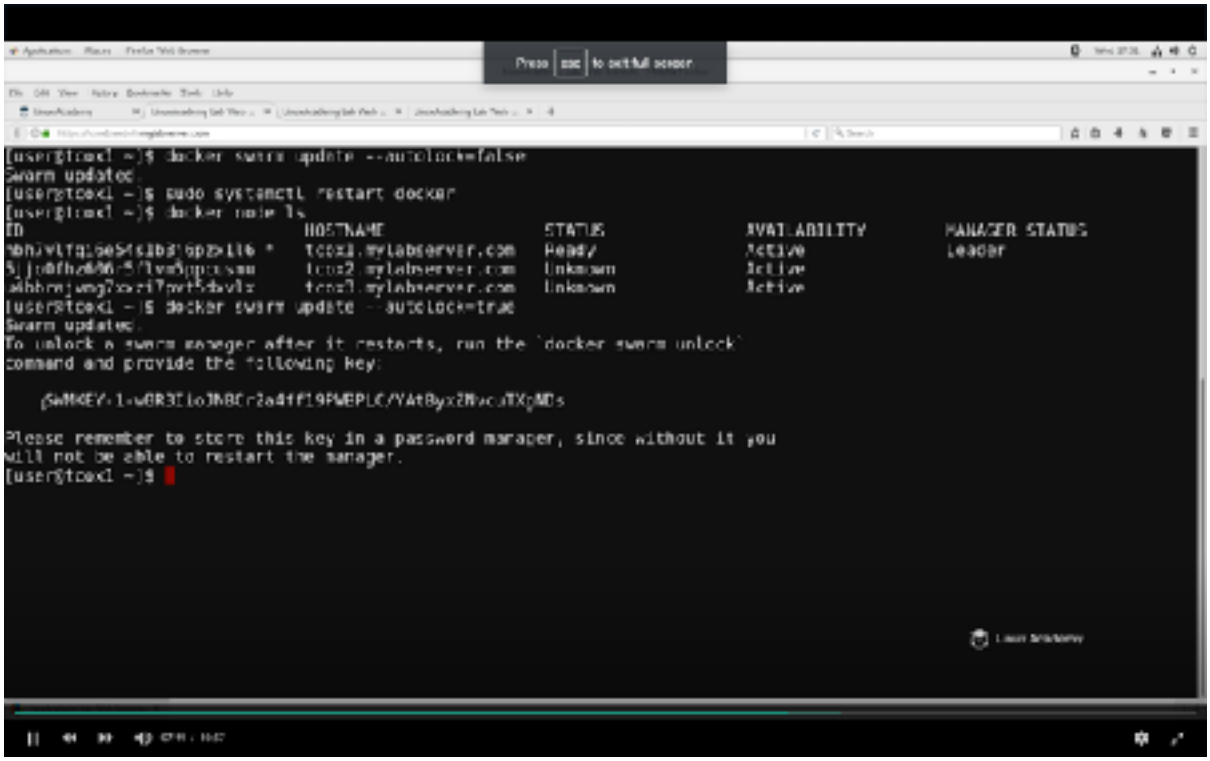
Services

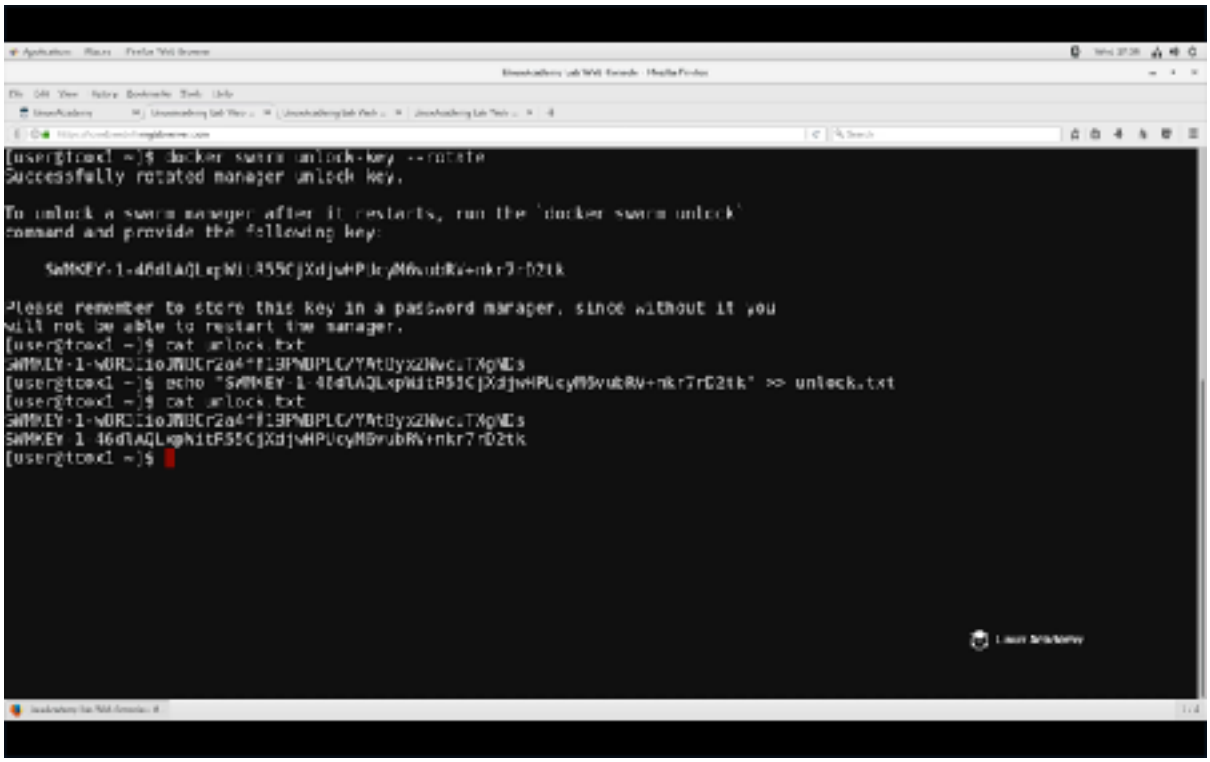
- Encapsulate an application or function.
- Can run on 1 to n nodes at any time.
- Functionality is easily accessed using features like 'routing mesh' outside the worker nodes.
- Multiple instances are set to launch in a single command.
- Highly available clusters are available.
- Easily scalable (up or down as needed)

```
[root@bakumar1c ~]# docker ps
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS              PORTS
NAMES
89034f0b0858        docker/ucp-agent:2.2.4         "/bin/ucp-agent agent"   5 minutes ago       Up 2 minutes        2376/tcp
ucp-agent.3b4nswbkipb25ir9f7ynbatlx.hhqvf5nytmdhj6fdr4ve6iuzb
6a45cfcb0725        docker/ucp-controller:2.2.4    "/bin/controller ser…"   12 hours ago        Up 5 minutes (unhealthy)   0.0.0.0:443->8080/tcp
ucp-controller
fca953b8b1f5        docker/ucp-auth:2.2.4          "/usr/local/bin/enzi…"   12 hours ago        Up 2 minutes (healthy)     0.0.0.0:12385->4443/tcp
ucp-auth-api
07e3c5da3cbf        docker/ucp-swarm:2.2.4         "/bin/swarm manage -…"   12 hours ago        Up 5 minutes (unhealthy)   0.0.0.0:2376->2375/tcp
ucp-swarm-manager
f2dcc48ea923        docker/ucp-metrics:2.2.4       "/bin/entrypoint.sh …"   12 hours ago        Up 5 minutes (unhealthy)   0.0.0.0:12387->12387/tcp
ucp-metrics
38a8b17b16f5        docker/ucp-auth:2.2.4          "/usr/local/bin/enzi…"   12 hours ago        Up 2 minutes (healthy)     0.0.0.0:12386->4443/tcp
ucp-auth-worker
d79571690fb5        docker/ucp-auth-store:2.2.4    "rethinkdb --bind al…"   12 hours ago        Up 5 minutes (unhealthy)   0.0.0.0:12383-12384->12383-12384/tcp
ucp-auth-store
f058daf47434        docker/ucp-etcd:2.2.4          "/bin/etcd --data-di…"   12 hours ago        Up 5 minutes (unhealthy)   2380/tcp, 4001/tcp, 7001/tcp, 0.0.0.0:12380->12380/
tcp, 0.0.0.0:12379->2379/tcp   ucp-kv
e67c2fc06320        docker/ucp-cfssl:2.2.4         "/bin/ucp-ca serve -…"   12 hours ago        Up 5 minutes (healthy)     0.0.0.0:12381->12381/tcp
ucp-cluster-root-ca
f771d4adc096        docker/ucp-cfssl:2.2.4         "/bin/ucp-ca serve -…"   12 hours ago        Up 5 minutes (healthy)     0.0.0.0:12382->12382/tcp
ucp-client-root-ca
ec10ec0d72bf        docker/ucp-agent:2.2.4         "/bin/ucp-agent prox…"   12 hours ago        Up 5 minutes (unhealthy)   0.0.0.0:12376->2376/tcp
ucp-proxy
[root@bakumar1c ~]#
[root@bakumar1c ~]#
[root@bakumar1c ~]# docker node ls
ID                           HOSTNAME                   STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
3b4nswbkipb25ir9f7ynbatlx *   bakumar1c.mylabserver.com   Ready               Active              Leader              19.03.3
kymk0yl6ba9glifm9u0lv5o22     bakumar2c.mylabserver.com   Ready               Active                                  19.03.3
t3d4asxfobykzl2jpo3c6ddui     bakumar3c.mylabserver.com   Ready               Active                                  19.03.3
[root@bakumar1c ~]#
[root@bakumar1c ~]#
[root@bakumar1c ~]# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE                               PORTS
43u1mkj91rex        backUP_App          replicated          2/2                 httpd:latest                        *:80->80/tcp
n8k6a8f1s7ob        ucp-agent           global              3/3                 docker/ucp-agent:2.2.4
l98sucqljoab        ucp-agent-s390x     global              0/0                 docker/ucp-agent-s390x:2.2.4
n6wx8z1ojeux        ucp-agent-win       global              0/0                 docker/ucp-agent-win:2.2.4
```

Locking swarm

```
[root@bakumar1c ~]# docker swarm update --autolock=true
Swarm updated.
To unlock a swarm manager after it restarts, run the `docker swarm unlock`
command and provide the following key:

   SWMKEY-1-ivI60fro8gI//KIXlBZk7fcfQgeWFdRLqUAnSXvWQfk

Please remember to store this key in a password manager, since without it you
will not be able to restart the manager.

 [root@bakumar1c ~]# vim /tmp/unlock.txt
~]# systemctl stop docker
[root@bakumar1c ~]# systemctl start docker

[root@bakumar1c ~]#  docker node ls
Error response from daemon: Swarm is encrypted and needs to be unlocked before it can be
used. Please use "docker swarm unlock" to unlock it.
 [root@bakumar1c ~]# cat /tmp/unlock.txt
SWMKEY-1-ivI60fro8gI//KIXlBZk7fcfQgeWFdRLqUAnSXvWQfk
[root@bakumar1c ~]# docker swarm unlock
Please enter unlock key: <Provide above key>
[root@bakumar1c ~]#  docker node ls
ID                          HOSTNAME                    STATUS          AVAILABILITY      MANAGER STATUS     ENGINE VERSION
3b4nswbkipb25ir9f7ynbatlx *   bakumar1c.mylabserver.com   Ready           Active            Leader             19.03.3
kymk0yl6ba9glifm9u0lv5o22     bakumar2c.mylabserver.com   Ready           Active                               19.03.3
t3d4asxfobykzl2jpo3c6ddui     bakumar3c.mylabserver.com   Ready           Active                               19.03.3
```





Docker UCP is taking lot of system resources, stopping these services

```
[root@bakumar1c ~]# docker ps
CONTAINER ID        IMAGE                       COMMAND                  CREATED             STATUS              PORTS
NAMES
4358cd0549a2        httpd                       "httpd-foreground"       2 minutes ago       Up 2 minutes        80/tcp
testWeb
1d648fa54802        docker/ucp-agent:2.2.4      "/bin/ucp-agent agent"   12 minutes ago      Up 11 minutes       2376/tcp
ucp-agent.3b4nswbkipb25ir9f7ynbatlx.37tm2ijdo3qo85r2mh0jyk5gu
6a45cfcb0725        docker/ucp-controller:2.2.4 "/bin/controller ser…"   13 hours ago        Up 12 minutes (healthy)   0.0.0.0:443->8080/
tcp                                             ucp-controller
fca953b8b1f5        docker/ucp-auth:2.2.4       "/usr/local/bin/enzi…"   13 hours ago        Up 13 minutes (healthy)   0.0.0.0:12385->4443/
tcp                                             ucp-auth-api
07e3c5da3cbf        docker/ucp-swarm:2.2.4      "/bin/swarm manage -…"   13 hours ago        Up 13 minutes (healthy)   0.0.0.0:2376-
>2375/tcp                                       ucp-swarm-manager
f2dcc48ea923        docker/ucp-metrics:2.2.4    "/bin/entrypoint.sh …"   13 hours ago        Up 13 minutes (healthy)   0.0.0.0:12387-
>12387/tcp                                      ucp-metrics
38a8b17b16f5        docker/ucp-auth:2.2.4       "/usr/local/bin/enzi…"   13 hours ago        Up 13 minutes (healthy)   0.0.0.0:12386->4443/
tcp                                             ucp-auth-worker
d79571690fb5        docker/ucp-auth-store:2.2.4 "rethinkdb --bind al…"   13 hours ago        Up 13 minutes (healthy)
0.0.0.0:12383-12384->12383-12384/tcp            ucp-auth-store
f058daf47434        docker/ucp-etcd:2.2.4       "/bin/etcd --data-di…"   13 hours ago        Up 13 minutes (healthy)   2380/tcp, 4001/tcp,
7001/tcp, 0.0.0.0:12380->12380/tcp, 0.0.0.0:12379->2379/tcp   ucp-kv
e67c2fc06320        docker/ucp-cfssl:2.2.4      "/bin/ucp-ca serve -…"   13 hours ago        Up 13 minutes (healthy)   0.0.0.0:12381-
>12381/tcp                                      ucp-cluster-root-ca
f771d4adc096        docker/ucp-cfssl:2.2.4      "/bin/ucp-ca serve -…"   13 hours ago        Up 13 minutes (healthy)   0.0.0.0:12382-
>12382/tcp                                      ucp-client-root-ca
ec10ec0d72bf        docker/ucp-agent:2.2.4      "/bin/ucp-agent prox…"   13 hours ago        Up 13 minutes (healthy)   0.0.0.0:12376-
>2376/tcp                                       ucp-proxy
[root@bakumar1c ~]#
[root@bakumar1c ~]# docker container run --rm -it \
>       --name ucp \
>       -v /var/run/docker.sock:/var/run/docker.sock \
>       docker/ucp \
>       stop
```

Deleting docker processes

```
[root@bakumar1c ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@bakumar1c ~]#
[root@bakumar1c ~]#
[root@bakumar1c ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
fe12c2938fb7        httpd:latest        "httpd-foreground"   15 hours ago        Dead                                    backUP_App.2.w3219zrf9qu54k3x9zkmxddgs
a74d74428e39        httpd               "httpd-foreground"   21 hours ago        Exited (0) 15 hours ago                 testSyslog
113d82c4ec74        httpd               "httpd-foreground"   21 hours ago        Exited (0) 15 hours ago                 testApp
[root@bakumar1c ~]# docker rm testApp
testApp
[root@bakumar1c ~]# docker rm testSyslog
testSyslog
[root@bakumar1c ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
fe12c2938fb7        httpd:latest        "httpd-foreground"   15 hours ago        Dead                                    backUP_App.2.w3219zrf9qu54k3x9zkmxddgs

[root@bakumar1c ~]# docker rm backUP_App.2.w3219zrf9qu54k3x9zkmxddgs
backUP_App.2.w3219zrf9qu54k3x9zkmxddgs
[root@bakumar1c ~]#
[root@bakumar1c ~]#
[root@bakumar1c ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@bakumar1c ~]#
[root@bakumar1c ~]#
[root@bakumar1c ~]#
[root@bakumar1c ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@bakumar1c ~]#
[root@bakumar1c ~]#
[root@bakumar1c ~]# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
43u1mkj91rex        backUP_App          replicated          2/2                 httpd:latest        *:80->80/tcp

 [root@bakumar1c ~]# docker node ls
ID                            HOSTNAME                    STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
3b4nswbkipb25ir9f7ynbatlx *   bakumar1c.mylabserver.com   Ready               Active              Leader              19.03.3
kymk0yl6ba9glifm9u0lv5o22     bakumar2c.mylabserver.com   Ready               Active                                  19.03.3
t3d4asxfobykzl2jpo3c6ddui     bakumar3c.mylabserver.com   Ready               Active                                  19.03.3
[root@bakumar1c ~]#
```

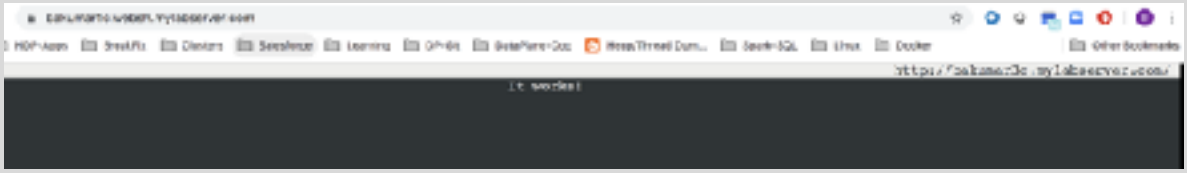
Exposing ports to all nodes/containers

```
[root@bakumar1c ~]# docker service create --name testWeb --publish 80:80 httpd

[root@bakumar1c ~]#
[root@bakumar1c ~]# docker service ps testWeb
ID             NAME          IMAGE          NODE                  DESIRED STATE    CURRENT STATE            ERROR
PORTS
nxq7on1ecwm1   testWeb.1     httpd:latest   bakumar1c.mylabserver.com   Running          Running about a minute ago
[root@bakumar1c ~]# docker service ls
ID             NAME          MODE          REPLICAS      IMAGE          PORTS
c7s4qyjwyuc1   testWeb       replicated    1/1           httpd:latest   *:80->80/tcp
[root@bakumar1c ~]# docker node ls
ID                            HOSTNAME                  STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
3b4nswbkipb25ir9f7ynbatlx *   bakumar1c.mylabserver.com   Ready     Active         Leader           19.03.3
kymk0yl6ba9glifm9u0lv5o22     bakumar2c.mylabserver.com   Ready     Active                          19.03.3
t3d4asxfobykzl2jpo3c6ddui     bakumar3c.mylabserver.com   Ready     Active                          19.03.3
[root@bakumar1c ~]#
```

Try to reachout this from any host now.

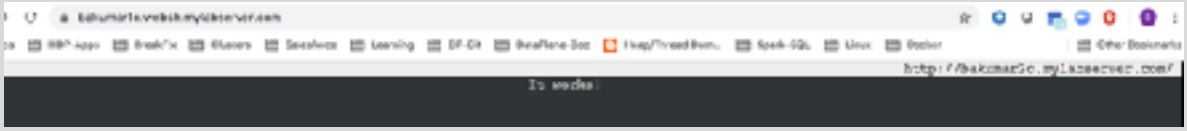
[root@bakumar1c ~]# elinks bakumar1c.mylabserver.com

"It works"



[root@bakumar1c ~]# elinks bakumar2c.mylabserver.com

"it works"



Similarly for node3:

[root@bakumar1c ~]# elinks bakumar3c.mylabserver.com

"It works"

Service scaling up

For now its 1 replica

```
[root@bakumar1c ~]# docker service ls
ID             NAME        MODE         REPLICAS     IMAGE          PORTS
c7s4qyjwyuc1   testWeb     replicated   1/1          httpd:latest   *:80->80/tcp
```

Scaling up to 3 replica's on the fly without stopping service:

```
[root@bakumar1c ~]# docker service update --replicas 3 testWeb
testWeb
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
[root@bakumar1c ~]#
[root@bakumar1c ~]# docker service ls
ID             NAME        MODE         REPLICAS     IMAGE          PORTS
c7s4qyjwyuc1   testWeb     replicated   3/3          httpd:latest   *:80->80/tcp
[root@bakumar1c ~]#
```

```
[root@bakumar1c ~]#
[root@bakumar1c ~]# docker service ps testWeb
ID             NAME        IMAGE          NODE                    DESIRED STATE   CURRENT STATE           ERROR
PORTS
nxq7on1ecwm1   testWeb.1   httpd:latest   bakumar1c.mylabserver.com   Running         Running 6 minutes ago
7yhep1jlryex   testWeb.2   httpd:latest   bakumar3c.mylabserver.com   Running         Running 2 minutes ago
s8f8fbf97jjv   testWeb.3   httpd:latest   bakumar2c.mylabserver.com   Running         Running 2 minutes ago
[root@bakumar1c ~]#
```

In busy cluster it may take time to replica but not immediately.

In 3 node cluster, we can scale to any number and down to any number , by default it is detach mode.

```
[root@bakumar1c ~]#
[root@bakumar1c ~]# docker service update --replicas 10 testWeb
testWeb
overall progress: 10 out of 10 tasks
1/10: running   [==================================================>]
2/10: running   [==================================================>]
3/10: running   [==================================================>]
4/10: running   [==================================================>]
5/10: running   [==================================================>]
6/10: running   [==================================================>]
7/10: running   [==================================================>]
8/10: running   [==================================================>]
9/10: running   [==================================================>]
10/10: running   [==================================================>]
verify: Service converged
[root@bakumar1c ~]# docker service ls
ID             NAME        MODE         REPLICAS     IMAGE          PORTS
c7s4qyjwyuc1   testWeb     replicated   10/10        httpd:latest   *:80->80/tcp
[root@bakumar1c ~]# docker service ps testWeb
ID             NAME        IMAGE          NODE                    DESIRED STATE   CURRENT STATE           ERROR
PORTS
nxq7on1ecwm1   testWeb.1   httpd:latest   bakumar1c.mylabserver.com   Running         Running 10 minutes ago
7yhep1jlryex   testWeb.2   httpd:latest   bakumar3c.mylabserver.com   Running         Running 5 minutes ago
s8f8fbf97jjv   testWeb.3   httpd:latest   bakumar2c.mylabserver.com   Running         Running 5 minutes ago
```

Limiting Resource like Cpu and Memory:
Limit and reservation:

--limit-cpu
--limit-memory (Should not less than 4 MB)
--reserve-cpu(Should not exceed physically available CPU's)
--reserve-memory(Should not exceed physically available memory)

For example

[root@bakumar1c ~]# docker service **update --limit-cpu**=0.5 **--reserve-cpu**=0.75 **--limit-memory**=64m **--reserve-memory**=128m testWeb
testWeb
overall progress: 3 out of 3 tasks
1/3: running [==========>]
2/3: running [==========>]
3/3: running [==========>]
verify: Service converged
[root@bakumar1c ~]# docker service ps testWeb

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE	ERROR	PORTS
z4hf51m1vn4r	testWeb.1	httpd:latest	bakumar1c.mylabserver.com	Running	Running 2 minutes ago		
nxq7on1ecwm1	_ testWeb.1	httpd:latest	bakumar1c.mylabserver.com	Shutdown	Shutdown 2 minutes ago		
9d99r1f6ldld	testWeb.2	httpd:latest	bakumar3c.mylabserver.com	Running	Running 2 minutes ago		
7yhep1jlryex	_ testWeb.2	httpd:latest	bakumar3c.mylabserver.com	Shutdown	Shutdown 2 minutes ago		
i7qmangys4pk	testWeb.3	httpd:latest	bakumar2c.mylabserver.com	Running	Running 2 minutes ago		
pffqwm67vtkk	_ testWeb.3	httpd:latest	bakumar2c.mylabserver.com	Shutdown	Shutdown 2 minutes ago		

[root@bakumar1c ~]#

When we update cpu's and memory, it ll shutdown existing containers and start new containers with newly updated resources.

For replica's update it wont shutdown.

Increase or decrease Number of replicas

```
[root@bakumar1c ~]# docker service ls
ID             NAME           MODE           REPLICAS       IMAGE          PORTS
[root@bakumar1c ~]#
[root@bakumar1c ~]#
[root@bakumar1c ~]# docker service create --name testWeb -p 80:80 httpd
umo8yv58ym2e2iv284le30cyh
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
[root@bakumar1c ~]#
[root@bakumar1c ~]#
[root@bakumar1c ~]# docker service ls
ID             NAME           MODE           REPLICAS       IMAGE          PORTS
umo8yv58ym2e   testWeb        replicated     1/1            httpd:latest   *:80->80/tcp
[root@bakumar1c ~]#
[root@bakumar1c ~]# docker service ps testWeb
ID             NAME           IMAGE          NODE                   DESIRED STATE   CURRENT STATE           ERROR
PORTS
oc7o0ni0puke   testWeb.1      httpd:latest   bakumar1c.mylabserver.com   Running         Running 28 seconds ago
[root@bakumar1c ~]# docker service update --replicas 3 --detach=false testWeb
testWeb
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
[root@bakumar1c ~]# docker service ps testWeb
ID             NAME           IMAGE          NODE                   DESIRED STATE   CURRENT STATE           ERROR
PORTS
oc7o0ni0puke   testWeb.1      httpd:latest   bakumar1c.mylabserver.com   Running         Running about a minute ago
vcobnyx1s1nc   testWeb.2      httpd:latest   bakumar2c.mylabserver.com   Running         Running 8 seconds ago
xqe0m7r3oni7   testWeb.3      httpd:latest   bakumar3c.mylabserver.com   Running         Running 7 seconds ago
[root@bakumar1c ~]# docker service create --name testIngnx -p 5901:80 nginx
75m8eyd9umd9f6m1cchsgoeen
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
[root@bakumar1c ~]#
```

We can update/scale of number of replicas more than one service at once using “scale”
“Docker service scale <serviceName=>”

```
^C
[root@bakumaric ~]# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
15m8eyd9umd9        testIngnx           replicated          1/1                 nginx:latest        *:5901->80/tcp
mnc8yv58ymzb        testWeb             replicated          3/3                 httpd:latest        *:80->80/tcp
[root@bakumaric ~]# docker service update --replicas 3 --detach=false testWeb testnginx
"docker service update" requires exactly 1 argument.
See 'docker service update --help'.

Usage:  docker service update [OPTIONS] SERVICE

Update a service
[root@bakumaric ~]# docker service scale  --detach=false testWeb=3 testnginx=3
testWeb scaled to 3
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
testnginx: Error: No such service: testnginx
[root@bakumaric ~]# docker service scale  --detach=false testWeb=2 testIngnx=3
testWeb scaled to 2
testIngnx scaled to 3
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Waiting 5 seconds to verify that tasks are stable...
```

```
[root@bakumar1c ~]# docker service ls
ID             NAME          MODE          REPLICAS      IMAGE         PORTS
75m8eyd9umd9   testIngnx     replicated    1/1           nginx:latest  *:5901->80/tcp
umo8yv58ym2e   testWeb       replicated    3/3           httpd:latest  *:80->80/tcp
[root@bakumar1c ~]# docker service update --replicas 3 --detach=false testWeb testnginx
"docker service update" requires exactly 1 argument.
See 'docker service update --help'.

Usage:  docker service update [OPTIONS] SERVICE

Update a service
[root@bakumar1c ~]# docker service scale  --detach=false testWeb=3 testnginx=3
testWeb scaled to 3
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
testnginx: Error: No such service: testnginx
[root@bakumar1c ~]# docker service scale  --detach=false testWeb=2 testIngnx=3
testWeb scaled to 2
testIngnx scaled to 3
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Waiting 5 seconds to verify that tasks are stable...


[root@bakumar1c ~]# docker service ls
ID             NAME          MODE          REPLICAS      IMAGE         PORTS
75m8eyd9umd9   testIngnx     replicated    3/3           nginx:latest  *:5901->80/tcp
umo8yv58ym2e   testWeb       replicated    2/2           httpd:latest  *:80->80/tcp



[root@bakumar1c ~]# docker service ps testIngnx
ID             NAME          IMAGE         NODE                  DESIRED STATE   CURRENT STATE           ERROR
PORTS
m9vzu7vz6cqv   testIngnx.1   nginx:latest  bakumar1c.mylabserver.com   Running         Running 14 minutes ago
k9syzjvy183a   testIngnx.2   nginx:latest  bakumar3c.mylabserver.com   Running         Running 7 minutes ago
imgb34uabe5a   testIngnx.3   nginx:latest  bakumar2c.mylabserver.com   Running         Running 7 minutes ago
[root@bakumar1c ~]# docker service ps testWeb
ID             NAME          IMAGE         NODE                  DESIRED STATE   CURRENT STATE           ERROR
PORTS
oc7o0ni0puke   testWeb.1     httpd:latest  bakumar1c.mylabserver.com   Running         Running 16 minutes ago
vcobnyx1s1nc   testWeb.2     httpd:latest  bakumar2c.mylabserver.com   Running         Running 15 minutes ago
[root@bakumar1c ~]#
```

Running Replicated vs. Global Services

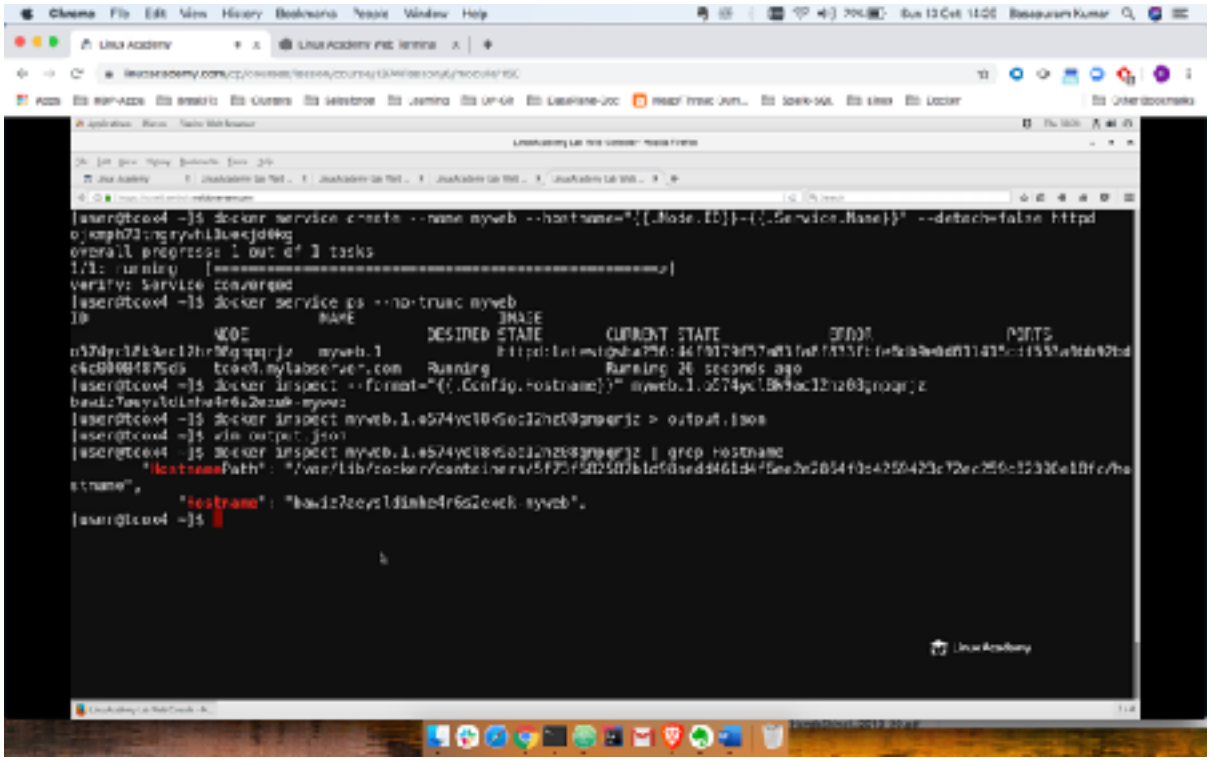
```
[root@bakumar1c ~]# docker service rm testWeb
testWeb
[root@bakumar1c ~]# docker service rm testIngnx
testIngnx
[root@bakumar1c ~]# docker service ls
ID             NAME           MODE           REPLICAS       IMAGE          PORTS
[root@bakumar1c ~]# docker service create --name testWeb -p 80:80 httpd
xjju62hs7up7y74409xgzmloj
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
[root@bakumar1c ~]#
[root@bakumar1c ~]# docker service ls
ID             NAME           MODE           REPLICAS       IMAGE          PORTS
xjju62hs7up7   testWeb        replicated     1/1            httpd:latest   *:80->80/tcp
[root@bakumar1c ~]# docker service ps testWeb
ID             NAME           IMAGE          NODE                    DESIRED STATE   CURRENT STATE           ERROR
PORTS
lf9zr13ck6xa   testWeb.1      httpd:latest   bakumar1c.mylabserver.com   Running         Running 19 seconds ago
[root@bakumar1c ~]# docker service scale testWeb=3
testWeb scaled to 3
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
[root@bakumar1c ~]# docker service ps testWeb
ID             NAME           IMAGE          NODE                    DESIRED STATE   CURRENT STATE           ERROR
PORTS
lf9zr13ck6xa   testWeb.1      httpd:latest   bakumar1c.mylabserver.com   Running         Running 55 seconds ago
s8eb8smbx774   testWeb.2      httpd:latest   bakumar2c.mylabserver.com   Running         Running 9 seconds ago
ard322q0xz8g   testWeb.3      httpd:latest   bakumar3c.mylabserver.com   Running         Running 8 seconds ago
[root@bakumar1c ~]# docker service ls
ID             NAME           MODE           REPLICAS       IMAGE          PORTS
xjju62hs7up7   testWeb        replicated     3/3            httpd:latest   *:80->80/tcp
[root@bakumar1c ~]#
```

Global services for replcation

```
[root@bakumar1c ~]# docker service create --name testIngnx -p 5901:80  --mode global --detach=false nginx
qntwaq3744zxcchd6aajpts5z
overall progress: 3 out of 3 tasks
3b4nswbkipb2: running   [==================================================>]
t3d4asxfobyk: running   [==================================================>]
kymk0yl6ba9g: running   [==================================================>]
verify: Service converged
[root@bakumar1c ~]# docker service ls
ID             NAME          MODE          REPLICAS        IMAGE          PORTS
qntwaq3744zx   testIngnx     global        3/3             nginx:latest   *:5901->80/tcp
xjju62hs7up7   testWeb       replicated    3/3             httpd:latest   *:80->80/tcp
[root@bakumar1c ~]# docker service ps testIngnx
ID             NAME                     IMAGE          NODE                 DESIRED STATE   CURRENT STATE
 ERROR          PORTS
i1v6hvnf8lot   testIngnx.t3d4asxfobykzl2jpo3c6ddui   nginx:latest   bakumar3c.mylabserver.com   Running         Running 39 seconds ago

7gst1f2alr9y   testIngnx.kymk0yl6ba9glifm9u0lv5o22   nginx:latest   bakumar2c.mylabserver.com   Running         Running 41 seconds ago

7ifh0ddsxh2j   testIngnx.3b4nswbkipb25ir9f7ynbatlx   nginx:latest   bakumar1c.mylabserver.com   Running         Running 39 seconds ago

[root@bakumar1c ~]# docker service update  --replicas 3 testIngnx
replicas can only be used with replicated mode
[root@bakumar1c ~]# docker service scale testIngnx=2
testIngnx: scale can only be used with replicated mode
[root@bakumar1c ~]#
```

Global service replication wont support updating manually using scale/"update replicas", when system/cluster scales, it will update one more replica automatically and when system down, it will scaledown automatically. That is number of nodes == number of global replica copies.

Templates:



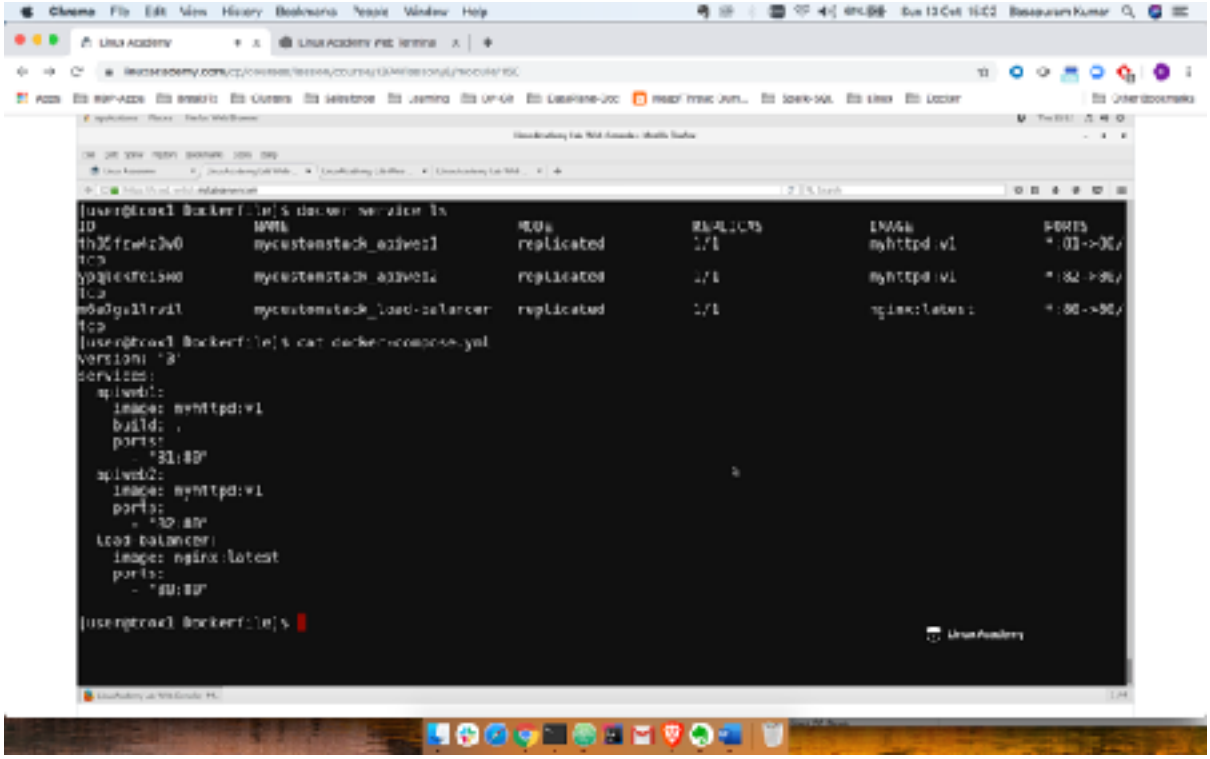
```
[root@bakumar1c ~]# yum install epel-release -y
[root@bakumar1c ~]# yum install python-pip -y
```

Docker container Inspect

```
[root@bakumar1c ~]# docker inspect TestApp
[root@bakumar1c ~]# docker  inspect TestApp | grep -i ipaddr
[root@bakumar1c ~]# docker container  inspect --format ="{{json .State}}" TestApp
[root@bakumar1c ~]# docker container  inspect --format ="{{json .NetworkSettings}}"
TestApp
```

```
[user@tcox1 Dockerfile]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
myhttpd             v1                  75ab1c52017f        2 minutes ago       322MB
nginx               latest              [illegible]         2 days ago          109MB
httpd               latest              [illegible]         2 days ago          [illegible]MB
centos              latest              3fa022595e1f        2 weeks ago         204MB
[user@tcox1 Dockerfile]$ docker rm testweb
Error response from daemon: You cannot remove a running container bds4faccac16308c2c275lac655cbc6a7705188d1c6ac9302a6d9
d45cd9c6afc. Stop the container before attempting removal or force remove
[user@tcox1 Dockerfile]$ docker stop testweb
testweb
[user@tcox1 Dockerfile]$ docker rm testweb
testweb
[user@tcox1 Dockerfile]$ docker rmi myhttpd:v1
Untagged: myhttpd:v1
Deleted: sha256:75ab1c52017f6a2c7589-5063213d0401bbcr74fccc5acdc9cc9c404c1af55b2
Deleted: sha256:501a8fcbae0734:c0490lac2[illegible]
Deleted: sha256:41a253332a[illegible]
Deleted: sha256:55e574845[illegible]
Deleted: sha256:[illegible]
Deleted: sha256:[illegible]
Deleted: sha256:[illegible]
[user@tcox1 Dockerfile]$
```

```
[user@tcox1 Dockerfile]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
        NAMES
d0ac7a0c41E7        myhttpd:v1          "/bin/sh -c 'apach..."   About a minute ago  Up About a minute   1.0.0.0:81->8
0/tcp   dockerfile_apiweb1_1
0407d14C3fce        myhttpd:v1          "/bin/sh -c 'apach..."   About a minute ago  Up About a minute   1.0.0.0:82->8
0/tcp   dockerfile_apiweb2_1
35f091c602e3        nginx:latest        "nginx -g 'daemon ..."   About a minute ago  Up About a minute   1.0.0.0:80->8
0/tcp   dockerfile_load-balancer_1
[user@tcox1 Dockerfile]$ docker-compose ps
          Name                        Command               State         Ports
-------------------------------------------------------------------------------------
dockerfile_apiweb1_1        /bin/sh -c apachectl -DFOR ...   Up      0.0.0.0:81->80/tcp
dockerfile_apiweb2_1        /bin/sh -c apachectl -DFOR ...   Up      0.0.0.0:82->80/tcp
dockerfile_load-balancer_1  nginx -g daemon off;             Up      0.0.0.0:80->80/tcp
[user@tcox1 Dockerfile]$
```



Docker Swarm – Service Troubleshooting

Docker Swarm – Service Troubleshooting

Whenever you are having issues with a service not deploying, there are several steps you can try (all of which you have seen throughout this course at one time or another):

- Docker node ls
  - Are all the nodes available and show that way in your cluster?
- Docker service ps [servicename]
  - Is the service partially running? Is the problem specific to the number of replicas?
- Docker service inspect [servicename] (with/without –format for output)
  - Did you apply labels and do they show? Did you deploy your service with a constraint but you mismatched the value with the one used on launch?

Docker Swarm – Service Troubleshooting

Whenever you are having issues with a service not deploying, there are several steps you can try (all of which you have seen throughout this course at one time or another):

- Docker ps
  - Is your cluster a 'locked' cluster that needs a key for operation (on restart)?
- Did you recently restore a swarm configuration and the expected service is not starting?
  - Make sure you re-initialize the new manager you restored to with the –force-new-cluster so that it is not attempting to contact previous nodes.
- Have you recently updated Docker?
  - Make sure you are using the official Docker repositories OR the setup script from Docker.com to avoid version issues or feature unavailability.

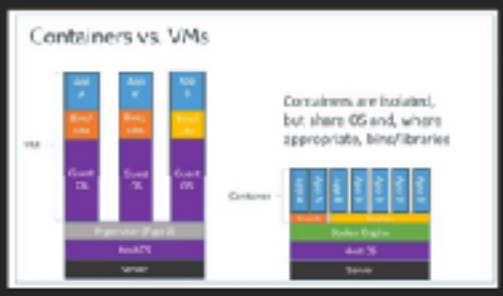
Troubleshooting Requires Experience

As in every technology, the ability to troubleshoot problems will rely on your experience with how the software behaves normally. In addition to Docker related issues, be sure you can rule out:

- SELinux issues – try 'setenforce 0' to put SELinux into 'Passive' mode and retry your task.
- Permissions – if you are launching a service service as an unprivileged user, be sure resources (like volumes/mounts) that you may be using are available to your UID/GID.
- CPU/Mem – when constraining your containers to a host, be sure it has the necessary resources to meet the needs of the service.
- Routing – make sure that end points using the service are on the same network segment or have the necessary routing to get there.
- Firewall – be sure the appropriate ports and protocols are enable to the destination IPs.

Dockerized Application Communication

Essentially, Docker containerized applications communicate with all other applications and infrastructure exactly the way you would expect any other application running in your environment with a few exceptions. Consider the Docker diagram below:



Side by side, the architecture of a Docker containerized application externally does not differ from what you might see on more traditional physical or virtual deployments.

However, there are a couple of caveats to keep in mind so that all of your applications know how to find each other.

Communication Caveats on Docker Containers

A few items to consider with Dockerized applications when putting them in place in your IT ecosystem:

- Routing
  - Either you will have to expose the Docker application/service via port redirection when you launch them OR you will have to provide a routing mechanism (statically) to the container network or their host(s).
- Port Redirection
  - As mentioned above, managing ports either through passing them directly through or redirecting them to the underlying host on different ports will affect how they behave in your environment.
- Portability
  - Making sure that data is external to the container application (on the host or via a network share) can have (sometimes significant) impacts on their performance

Containers Should Be...

Abstract

- You containerize an application so that it remains removed from other components in the stack. That abstraction makes updates easier, but does introduce additional potential areas that communication can fail.

Portable

- Applications that are containerized can be picked up and put almost anywhere and will be consistent in their content and behavior. However, re-establishing communication with other components in the stack will depend on the new location and how you have planned access to that resource.

Flexible

- Containers give you almost limitless flexibility. Be sure to try and refrain from launching your container services tied too closely to external (and changeable) variables (hard coded related IP addresses or hostnames not easily changed).

The Importance of Manager Nodes in a Swarm

Every Swarm will have 1 to N 'Manager' nodes in it. These special nodes are responsible for managing, directing, logging and reporting on the lifecycle of the Swarm in general.

The 'Raft Consensus Algorithm' is used to manage the swarm state. Using this 'consensus' method amongst the management nodes is designed to be sure that in the event of a failure of any manager, any other manager will have enough information stored on it to continue to operation the swarm as expected.

Raft tolerates up to (N-1)/2 failures and requires a majority (quorum) of (N/2)+1 to agree on any new instructions that are proposed to the cluster for execution.

Requirements

Breakdown of Manager Nodes for Fault Tolerance

Swarm Size	Majority	Fault Tolerance
1	1	0
2	2	0
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3
8	5	3
9	5	4

Requirements/Considerations

- Use Static IPs
- Immediately Replace Failed Managers
- Distribute Management Nodes for High Availability
- Monitor Swarm Health
- Have a Backup and Recovery Plan for the Swarm

Docker Storage and Volume

Storage Drivers

If you are using containers the way they are intended, you will be storing little to no data in a container's writable layer. Docker volumes are the more appropriate destination so that the container remains portable.

However, there are some cases where writing to the container storage layer is a necessity and you will need to choose an appropriate storage driver that is both performant and supported on your implementation platform. Consider the following when choosing:

1. Choose from a prioritized list (from Docker) for your platform/distribution (generally, the driver that requires the least amount of customization or configuration), followed by stability and performance
2. Be aware of the Docker edition you are using (CE or EE – will affect your available supported choices)
3. Some storage drivers may not be available for your current or desired filesystem types or your implementation platform (particularly Windows).

Docker Enterprise Edition

Here is a chart depicting which storage drivers are officially supported by the Docker Enterprise Edition implementation.

Distribution	Supported Storage Driver(s)
CentOS	devicemapper
Oracle Linux	devicemapper
Red Hat Enterprise Linux	devicemapper, overlay2
SUSE Linux Enterprise Server	btrfs
Ubuntu	aufs3

Docker Community Edition

There is less certainty around the drivers for Community Edition. Only some versions are tested actively, but the following are reported to work as expected for each distribution listed

Distribution	Supported Storage Driver(s)
CentOS	devicemapper, vfs
Oracle Linux	devicemapper, overlay2
Red Hat Enterprise Linux	devicemapper, overlay2
SUSE Linux Enterprise Server	btrfs
Debian/Ubuntu	aufs3, devicemapper, overlay2, overlay, vfs
Fedora	devicemapper, overlay2, overlay, vfs

Docker Community Edition (continued)

Whenever possible, using the 'overlay2' is recommended on Docker CE (which will be the default on installation)

NOTE: Docker for Mac and Windows is intended for development and not production use, as a result, the default drivers cannot be modified at this time.

Distribution	Supported Storage Driver(s)
Mac OSX/sierraOS	devicemapper (default, no customization is possible)
Windows	devicemapper (default, no customization is possible)

Storage Drivers – Workload Use Cases

Each driver will have performance characteristics that will make it more or less suited to certain workloads. *Although Docker considers all the storage drivers we are talking about to be 'stable', some are newer and under active development.*

Consider workloads as listed below for each driver:

Driver	Workload
aufs, overlay, overlay2	Operate at the file level (rather than the block level). This allows more efficient memory utilization but the container layer can grow quickly.
devicemapper, btrfs, zfs	Operate at the block level (rather than the file level). This allows better performance in write heavy workloads as the expense of more memory.
overlay	For workloads necessitating many small writes or containers with many layers or deep filesystems, overlay may perform better than overlay2.

Images and Layers

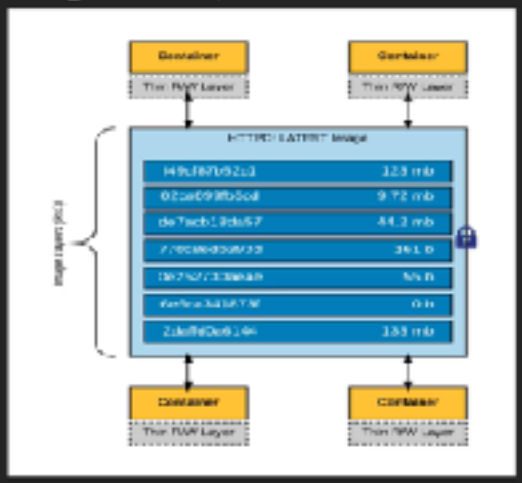
A Docker image is built up, from a series of layers, each representing a single instruction in the image's Dockerfile. Every layer except for the last is a read only layer.

A Dockerfile like:

```
FROM centos:latest
RUN yum update -y
RUN yum install -y telnet
ENTRYPOINT echo "This container has finished"
```

Would create four discrete layers in the construction of the image. Some commands can be chained so that multiple commands actually run, but only build a single layer ('RUN yum update -y && RUN yum install -y telnet' for example).

Images and Layers



In the case of the 'httpd:latest' image depicted to the left from Docker Hub, the image consists of 7 discrete layers.

Each layer represents a command that was run in the assembly of that image. Each layer is only the deltas from the previous image laying on top.

The only writable layer then becomes the 'Thin R/W Layer' for each container instantiated on that image's R/O layers.

Images and Layers

The configured Docker Storage Driver handles the details about how the image layers interact. See more information on Storage Driver Use Cases to determine which is supported and optimal for your platform, distribution and work load.

With each container having its own writable container layer, each image can maintain a 1 to N ratio of image storage to container storage (i.e. the image storage layers are never repeated).

As a result of Docker's storage strategy of 'Copy-On-Write', files and directories that exist in lower layers that are needed in higher layers, can be provided read access to them to avoid duplication. If that file needs to be modified, only then is it copied to the higher layer where the changes are stored.

Linux Academy | CloudAcademy

Containers and Deletion

Any time a container is deleted, any data that is written to the container read/write layer that is NOT stored in a data volume, will be deleted along with the container.

Data volumes, since they are NOT controlled by the storage driver (since they represent a file or directory on the host filesystem in the /var/lib/docker directory), are able to bypass the storage driver. As a result, their contents are not affected when a container is removed.

This allows containers to remain portable while providing a method of persistent storage outside of the image and container layered filesystem structure.

Linux Academy | CloudAcademy

Docker volumes

```
[root@bakumar1c ~]# docker volume ls
DRIVER              VOLUME NAME
[root@bakumar1c ~]# docker volume  create my_mount
my_mount
[root@bakumar1c ~]# docker volume ls
DRIVER              VOLUME NAME
local               my_mount
[root@bakumar1c ~]# docker volume inspect my_mount | more
[
    {
        "CreatedAt": "2019-10-13T11:15:28Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/my_mount/_data",
        "Name": "my_mount",
        "Options": {},
        "Scope": "local"
    }
]
```

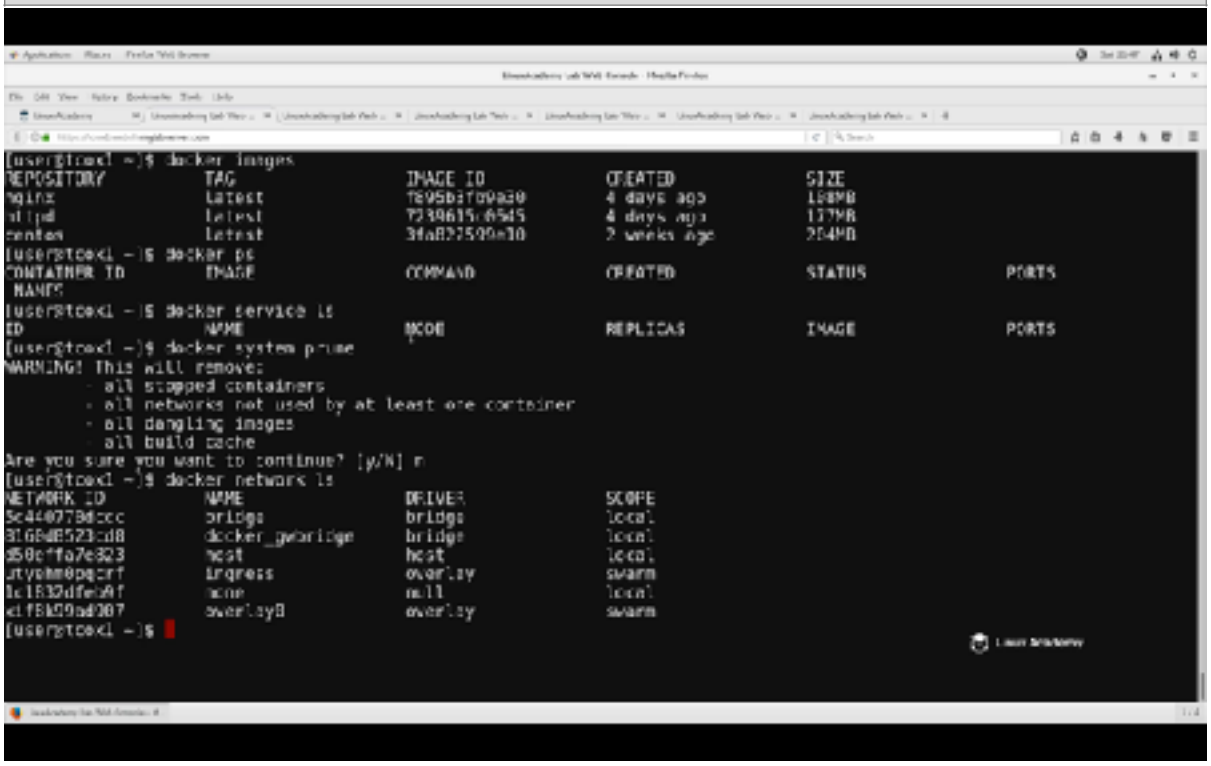
Check locally mounted volume:

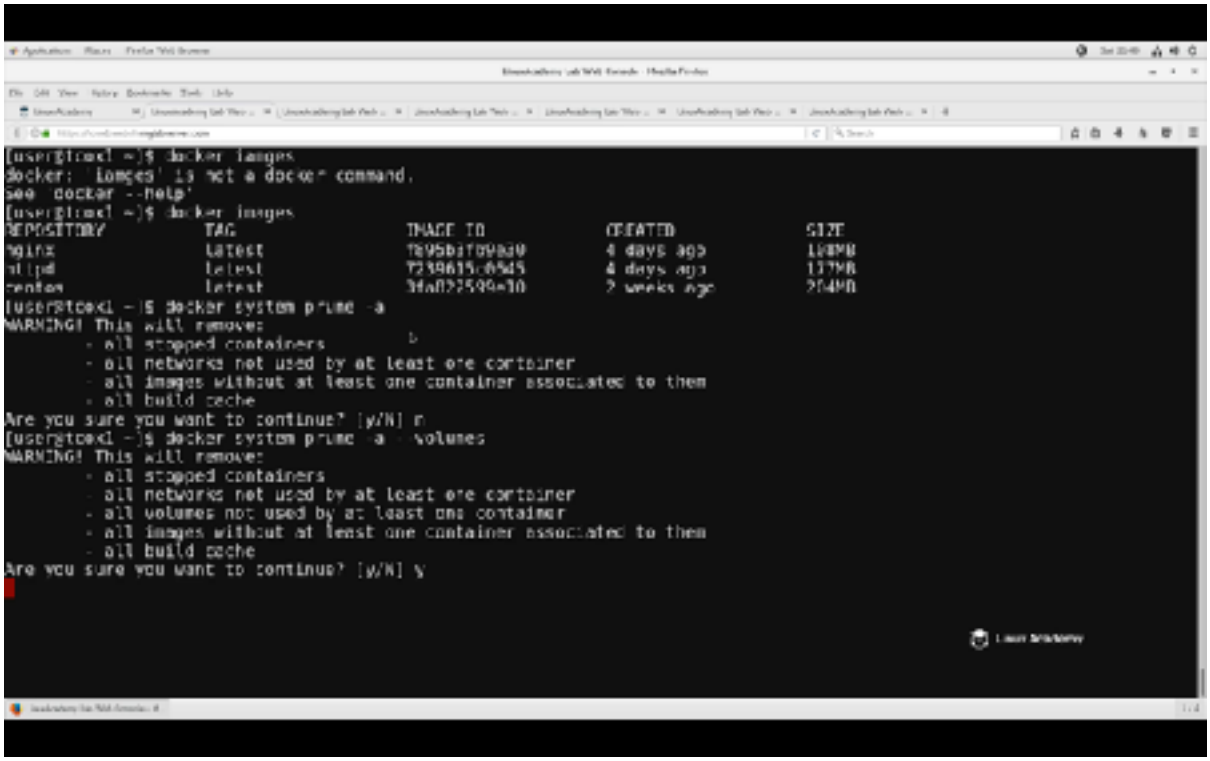
```
[root@bakumar1c ~]# cd /var/lib/docker/
[root@bakumar1c docker]# ls -lrt volumes/
total 32
drwxr-xr-x. 3 root root    18 Oct 13 11:15 my_mount
-rw-------. 1 root root 65536 Oct 13 11:15 metadata.db
[root@bakumar1c docker]# ls -lrt volumes/my_mount/
total 0
drwxr-xr-x. 2 root root 6 Oct 13 11:15 _data
[root@bakumar1c docker]# ls -lrt volumes/my_mount/_data/
total 0
[root@bakumar1c docker]#

[root@bakumar1c _data]#
[root@bakumar1c _data]# docker service  create  --name testWeb -p 80:80 --mount
source=my_mount,target=/internal-mount --detach=false --replicas 3 httpd
y47lshx0ybleodmw3vi655cyx
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
[root@bakumar1c _data]#
[root@bakumar1c _data]#
[root@bakumar1c _data]# pwd
/var/lib/docker/volumes/my_mount/_data
[root@bakumar1c _data]# ls -rlt
total 4
-rw-r--r--. 1 root root 27 Oct 13 11:19 hostfile.txt
 [root@bakumar1c _data]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
e0fefc4cabad        httpd:latest        "httpd-foreground"   3 minutes ago       Up 3 minutes        80/tcp
testWeb.1.q7ol0cajvl08uma5w8rqlw7oo
55500b2d46f6        httpd               "httpd-foreground"   About an hour ago   Up About an hour    80/tcp              TestApp
```

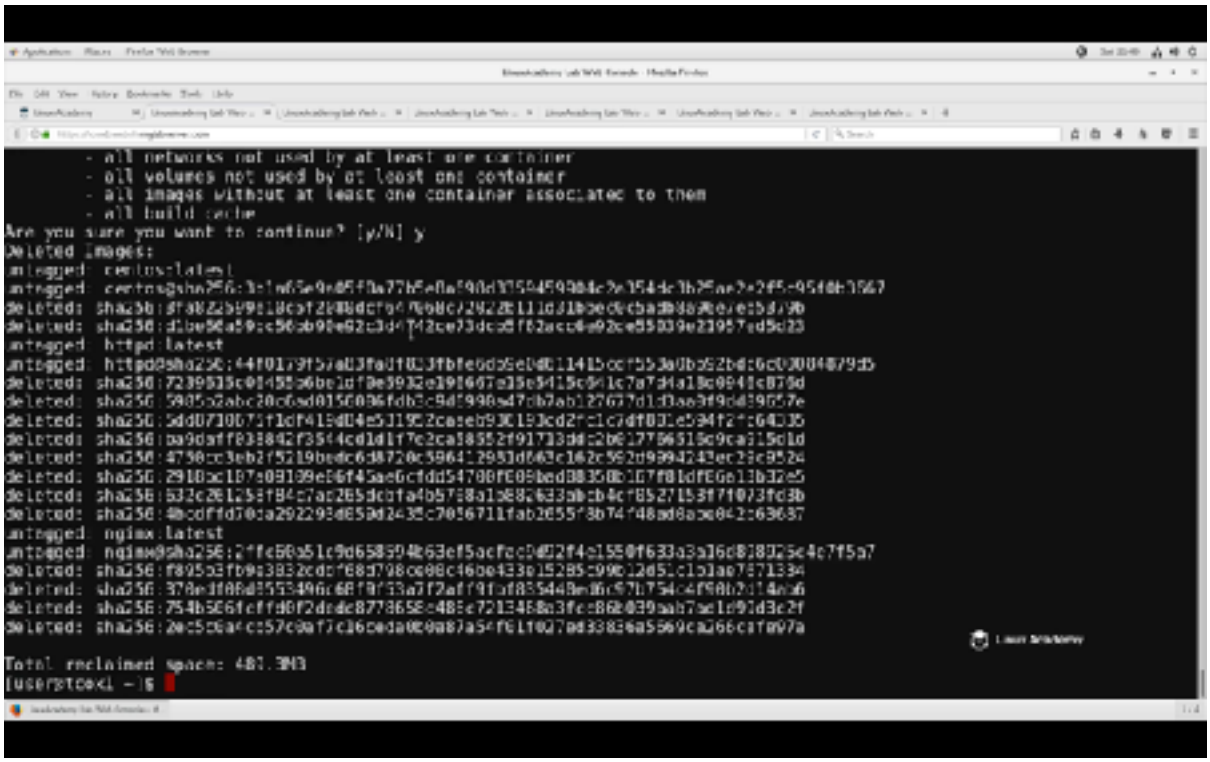
Logging into Docker container inside and check the shared mount directory(Target dir-/internal-mount)

Docker cleanup

```
[root@bakumar1c ~]# docker system prune --all --volumes
WARNING! This will remove:
  - all stopped containers
  - all networks not used by at least one container
  - all volumes not used by at least one container
  - all images without at least one container associated to them
  - all build cache

Are you sure you want to continue? [y/N] y
Deleted Images:
Total reclaimed space: 1.17GB
```







Docker Networking

```
[user@tcx6 ~]$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
5a8cbaec72b2        bridge              bridge              local
4e145Gd39f51        docker_gwbridge     bridge              local
d50e7fa7e323        host                host                local
mub5k7j8qm9k        ingress             overlay             swarm
1c1b32dfeb9f        none                null                local
[user@tcx6 ~]$ docker network create --driver=bridge --subnet=192.168.1.0/24 --opt "com.docker.network.driver.mtu"="15
00" devel0
1ed802a14586552a0d3df2c2a159ec4b0bdb0dcb0ebsb05ac0d04d511cfaa
[user@tcx6 ~]$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
5a8cbaec72b2        bridge              bridge              local
1ed802a14586        devel0              bridge              local
4e145Gd39f51        docker_gwbridge     bridge              local
d50e7fa7e323        host                host                local
mub5k7j8qm9k        ingress             overlay             swarm
1c1b32dfeb9f        none                null                local
[user@tcx6 ~]$
```

```
[user@tcx6 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
   NAMES
849b51484660        httpd               "http-foreground"   7 minutes ago       Up 7 minutes        0.0.0.0:80->80/tcp
   testweb
[user@tcx6 ~]$ docker network connect
"docker network connect" requires exactly 2 arguments.
See 'docker network connect --help'.

Usage:  docker network connect [OPTIONS] NETWORK CONTAINER

Connect a container to a network
[user@tcx6 ~]$ docker container inspect --format="{{.NetworkSettings.Networks.bridge.IPAddress}}" testweb
172.17.0.2
[user@tcx6 ~]$ docker network connect --ip=192.168.1.10 devel0 testweb
[user@tcx6 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
   NAMES
849b51484660        httpd               "http-foreground"   10 minutes ago      Up 9 minutes        0.0.0.0:80->80/tcp
   testweb
[user@tcx6 ~]$ docker container inspect --format="{{.NetworkSettings.Networks.bridge.IPAddress}}" testweb
172.17.0.2
[user@tcx6 ~]$ docker container inspect --format="{{.NetworkSettings.Networks.devel0.IPAddress}}" testweb
192.168.1.10
[user@tcx6 ~]$ elinks http://192.168.1.10
```

```
[user@tcox6 ~]$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
5a8cb4ec72b2        bridge              bridge              local
1ed882a14586        devel0              bridge              local
4e1456d90f51        docker_gwbridge     bridge              local
3500ffa7e623        host                host                local
mub5k7jfqm9k        ingress             overlay             swarm
1c1f132dfe59f       none                null                local
[user@tcox6 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
   NAMES
349b51434660        nttpd               "nttpd-foreground"  10 minutes ago      Up 10 minutes       0.0.0.0:80->80/tcp
   testweb
[user@tcox6 ~]$ docker network disconnect bridge testweb
[user@tcox6 ~]$ elinks http://192.168.1.10
```

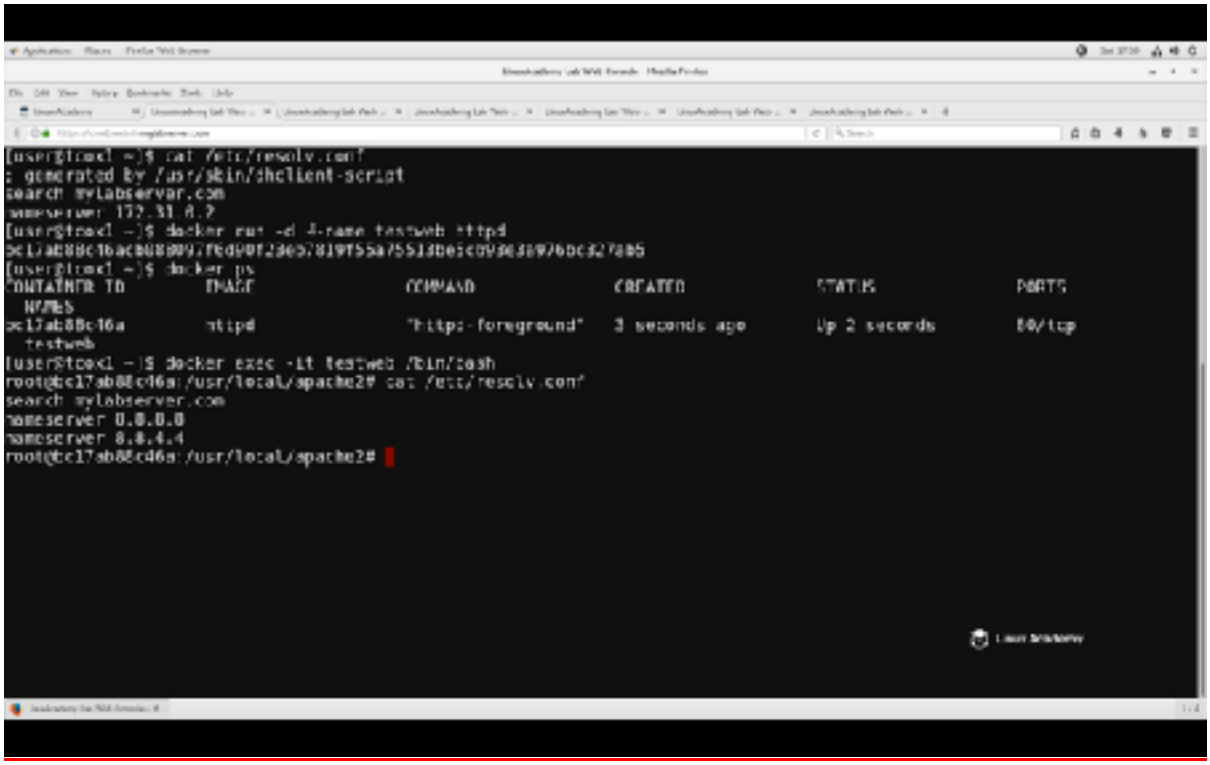
```
[user@tcox6 ~]$ docker container inspect --format="{{.NetworkSettings.Networks.bridge.IPAddress}}" testweb
<no value>
[user@tcox6 ~]$ docker container inspect --format="{{.NetworkSettings.Networks.devel0.IPAddress}}" testweb
192.168.1.10
[user@tcox6 ~]$
```

```
[user@tcxl ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              f495bif09930        4 days ago          188MB
ubipd               latest              7339615c6545        4 days ago          177MB
centos              latest              30a07500e30         2 weeks ago         204MB
[user@tcxl ~]$ cat /etc/resolv.conf
; generated by /usr/sbin/dhclient-script
search mylabserver.com
nameserver 172.31.0.2
[user@tcxl ~]$ docker run -d --name testweb ubipd
56f06650703ee7fc4cd9127bce111572db65cfeecd991fae57161e50b14c475e
[user@tcxl ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
  NAMES
56f06650703e        ubipd               "/usr/local/apache2/bin/apachectl -D FOREGROUND"   2 seconds ago       Up 1 second         80/tcp
  testweb
[user@tcxl ~]$ docker exec -it testweb /bin/bash
root@56f06650703e:/usr/local/apache2# cat /etc/resolv.conf
; generated by /usr/sbin/dhclient-script
search mylabserver.com
nameserver 172.31.0.2
root@56f06650703e:/usr/local/apache2# ping google.com
PING google.com (216.58.193.78): 56 data bytes
64 bytes from 216.58.193.78: icmp_seq=0 ttl=45 time=22.182 ms
64 bytes from 216.58.193.78: icmp_seq=1 ttl=45 time=22.283 ms
^C--- google.com ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 22.182/22.238/22.283/0.046 ms
root@56f06650703e:/usr/local/apache2#
```

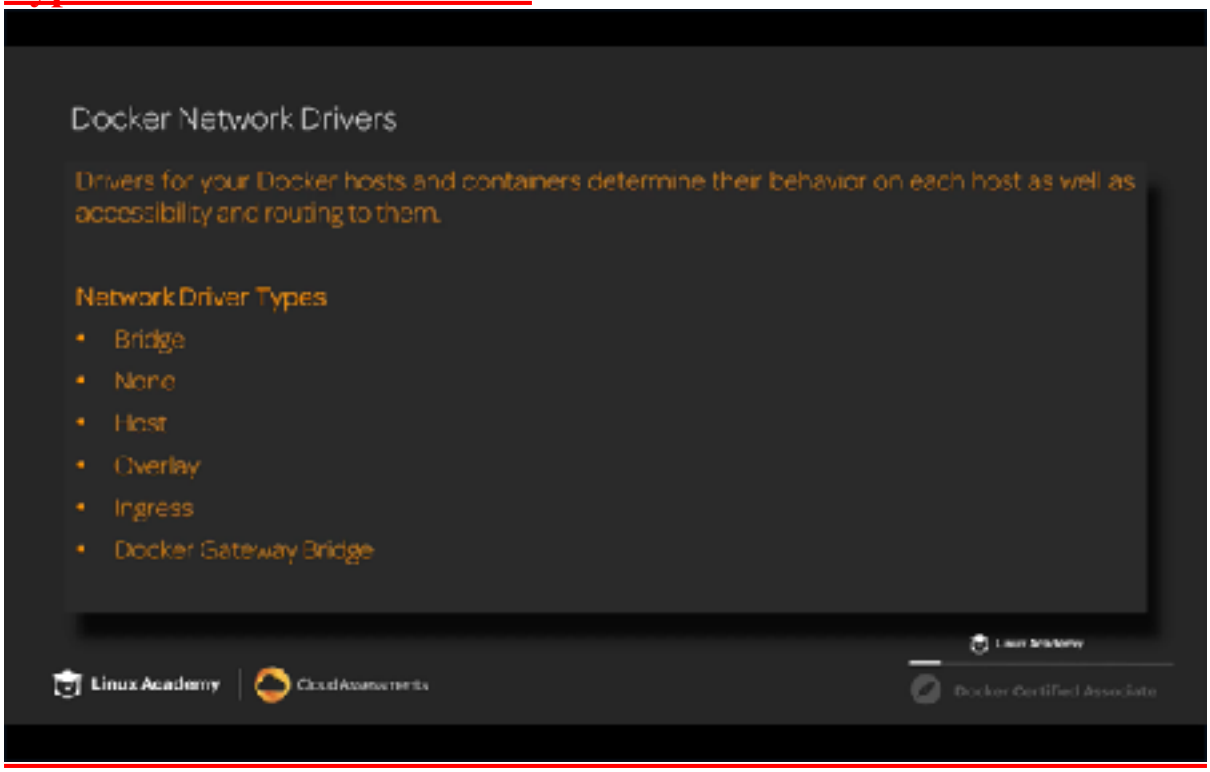
```
[user@tcxl ~]$ docker stop testweb
testweb
[user@tcxl ~]$ docker rm testweb
testweb
[user@tcxl ~]$ docker run -d --name testweb --dns=8.8.8.8 --dns=8.8.4.4 ubipd
8c847bf530aa7d4f1f4f72f0e0e3bfc1b4f0f2b6d4d4a82a2a06d5f4f05f6dc
[user@tcxl ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
  NAMES
8c847bf530aa        ubipd               "/usr/local/apache2/bin/apachectl -D FOREGROUND"   2 seconds ago       Up 1 second         80/tcp
  testweb
[user@tcxl ~]$ docker exec -it testweb /bin/bash
root@8c847bf530aa:/usr/local/apache2# cat /etc/resolv.conf
search mylabserver.com
nameserver 8.8.8.8
nameserver 8.8.4.4
root@8c847bf530aa:/usr/local/apache2#
```

```
[user@tcxl ~]$ docker stop testweb
testweb
[user@tcxl ~]$ docker rm testweb
testweb
[user@tcxl ~]$ docker run -d --name testweb --dns=8.8.8.8 --dns=8.8.4.4 httpd
9c847bf638aa7d4f1a9f172f0e0c5bfc1b4a0d2b8d4d45a82b2b0JE5T4V5V5JC
[user@tcxl ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
  NAMES
9c847bf638aa        httpd               "httpd-foreground"   2 seconds ago       Up 1 second         80/tcp
  testweb
[user@tcxl ~]$ docker exec -it testweb /bin/bash
root@9c847bf638aa:/usr/local/apache2# cat /etc/resolv.conf
search mylabserver.com
nameserver 8.8.8.8
nameserver 8.8.4.4
root@9c847bf638aa:/usr/local/apache2# exit
exit
[user@tcxl ~]$ docker stop testweb
testweb
[user@tcxl ~]$ docker rm testweb
testweb
[user@tcxl ~]$
```

```
        "dns": ["8.8.8.8", "8.8.4.4"]

-- INSERT --                                                    2,30-37       All
```



Types of Networks in Docker



Docker Network Drivers – Use Cases

Each Network Driver has a specific use case, in general:

None

- Used when the container(s) in question need absolutely no networking access at all.
- Containers operating on this driver can only be accessed on the host they are running on.
- These containers can be attached directly (using 'docker attach [containerid]') or executing by another command on the running container (using 'docker exec -it [containerid] [command]').
- Not commonly used.

Docker Network Drivers – Use Cases

Each Network Driver has a specific use case, in general:

Host

- Sometimes referred to as 'Host Only Networking'.
- Only accessible via the underlying host.
- Access to services can only be provided by exposing container service ports to the host system.

Docker Network Drivers – Use Cases

Each Network Driver has a specific use case, in general:

Overlay

- Allows communication among all Docker Daemons that are participating in a Swarm.
- It is a 'Swarm Scope' driver in that it extends itself (building previously non-existent networks on Workers if needed) to all daemons in the Swarm cluster
- Allows the communication of multiple services that may have replicas running on any number of Workers in the Swarm, regardless of their origination or destination.
- Default mode of Swarm communication.

Docker Network Drivers – Use Cases

Each Network Driver has a specific use case, in general:

Ingress

- Special overlay network that load balances network traffic amongst a given service's working nodes.
- Maintains a list of all IP addresses from nodes that participate in that service (using the IPVS module) and when a request comes in, routes to one of them for the indicated service.
- Provides the 'routing mesh' that allows services to be exposed to the external network without having a replica running on every node in the Swarm.

Docker Network Drivers – Use Cases

Each Network Driver has a specific use case, in general:

Docker Gateway Bridge

- Special bridge network that allows overlay networks (including Ingress) access to an individual Docker daemon's physical network.
- Every container run within a service is connected to the local Docker daemon's host network.
- Automatically created when a Swarm is initialized or joined.

Linux Academy | CloudAcademy

Docker Network Drivers – Port Publishing Mode

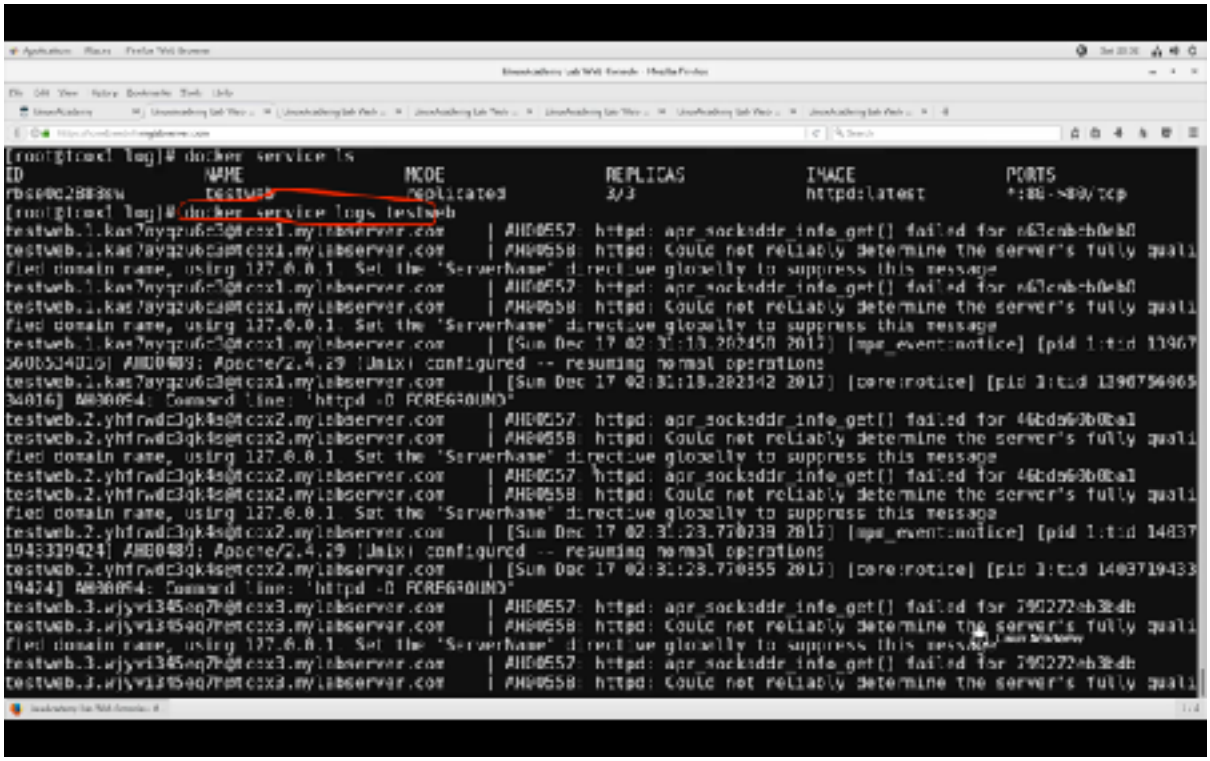
Two types of port publishing modes:

- Host
  - Ports for containers are only available on the underlying host system and are NOT available for services outside of the host where that instance exists.
  - Used in single host environments or in environments where you need complete control over routing.
  - You are responsible for knowing where all instances are at all times, controlled with "mode=host" in deployment.
- Ingress
  - Since it is responsible for 'routing mesh', it makes all published ports available on all hosts (nodes or Workers) participating in the cluster so that the service is accessible from every node regardless of whether there is a service replica running on it at any given time.

Linux Academy | CloudAcademy

Docker logs are stored in local machine

```
[root@bakumar1c ~]# ls -lrt  /var/log/messages*
-rw-------. 1 root root 5345031 Dec 13  2018 /var/log/messages-20181213
-rw-------. 1 root root 9650039 Jan 16  2019 /var/log/messages-20190116
-rw-------. 1 root root 2088501 Feb 13  2019 /var/log/messages-20190213
-rw-------. 1 root root 2858798 May  9 18:08 /var/log/messages-20190509
-rw-------. 1 root root 6490964 Oct 13 13:15 /var/log/messages
[root@bakumar1c ~]#

#docker service logs  testWeb
```



Docker deployment whether app

Create docker-compose.yml:

```
vi docker-compose.yml
```

docker-compose.yml

```
version: '3'
services:
  weather-app1:
    build:
      context: ./weather-app
      args:
        - VERSION=v2.0
    ports:
      - "8080:3000"
    networks:
     - weather_app
    environment:
      - NODE_ENV=production
  weather-app2:
    build:
      context: ./weather-app
      args:
        - VERSION=v2.0
    ports:
      - "8081:3000"
    networks:
     - weather_app
    environment:
      - NODE_ENV=production
  weather-app3:
    build:
      context: ./weather-app
      args:
        - VERSION=v2.0
    ports:
      - "8082:3000"
    networks:
     - weather_app
    environment:
      - NODE_ENV=production
```