Docker Administration

Maruthi S. Inukonda

09th Mar 2019

Agenda

- Containers
- Dockers
- Docker storage
- Docker Identity and Access management
- Docker networking
- Docker Resource (CPU, RAM) allocation
- Container security concerns.

Containers

What is a Container? (1/2) (recap)

- Linux Containers (LXC) is an operating-system-level virtualization method.
- For running multiple isolated Linux systems (containers) on a control host using a single Linux kernel.
- Directly runs on hardware. (No per-instruction level trapping)
- An unprivileged user on host can be privileged user on guest.
- Civil Engineering example :
    - Hostel complex having multiple rooms with shared resources.
      Viz., all the above but not study/bedroom.

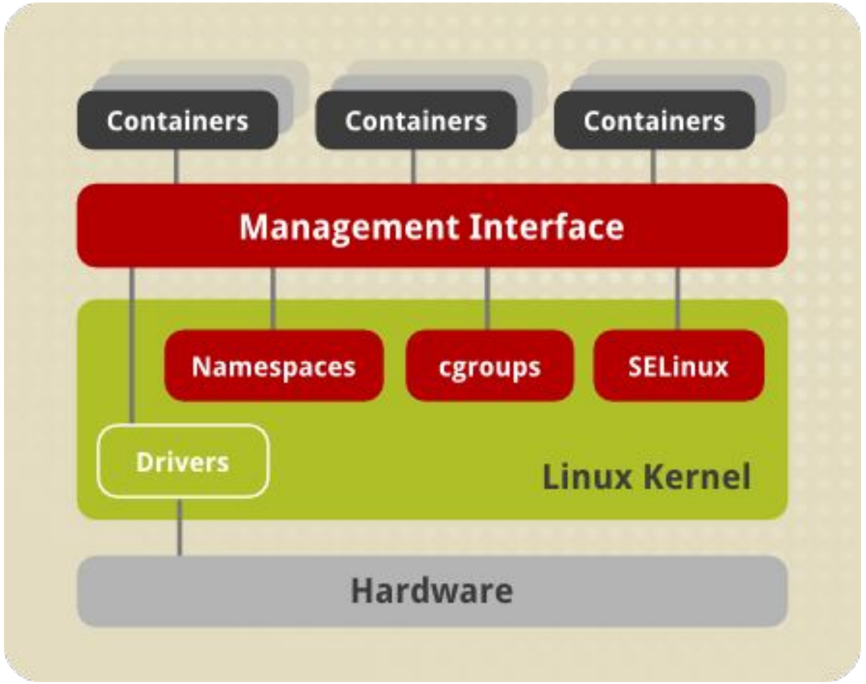


Image Courtesy: Redhat Customer Portal

What is a Container? (2/2) (recap)

It is implemented using following features in Linux

- Advanced Multi-layer Union FS (AUFS) or Overlay FS
- Kernel namespaces
- Cgroups
- Capabilities
- Netfilter, Netlink
- Bind mount
- Role-Based Access Control (RBAC)
  - Eg. SELinux, AppArmor

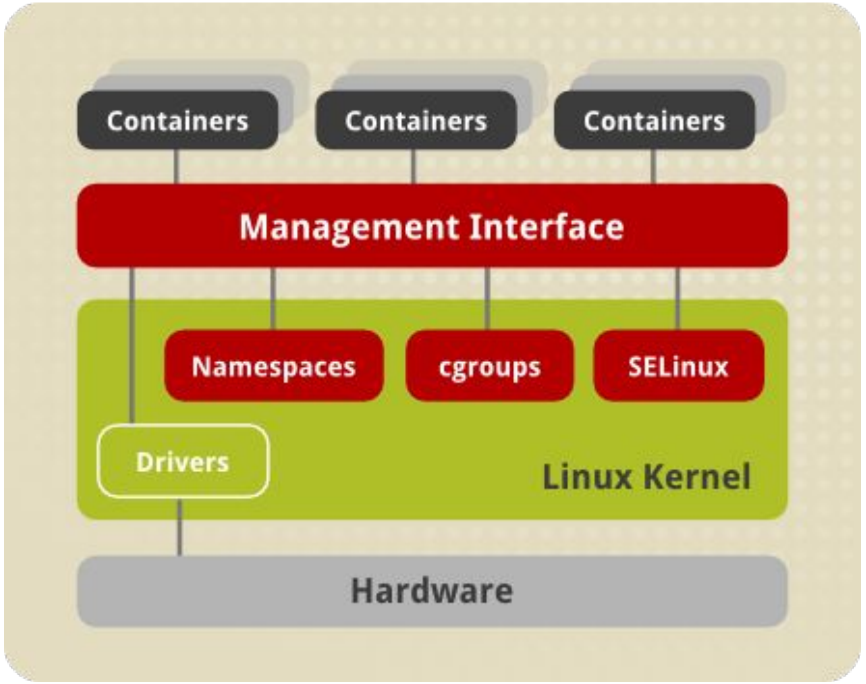


Image Courtesy: Redhat Customer Portal

Dockers

What is a Docker?

- Docker is a company that provides software (also called Docker) that allows you to build, run and manage software containers.
- It makes container deployment and administration quite easy.
- It allows re-use of containers created by others.
- It allows running multiple versions of application software with its dependencies on same host.

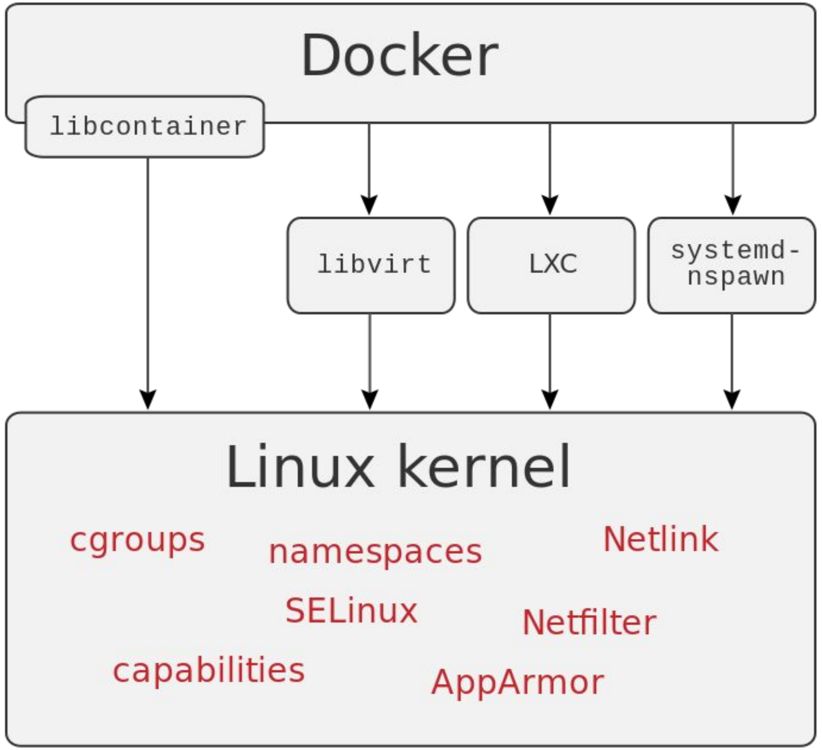


Image Courtesy: Maklaan's Docker blog post

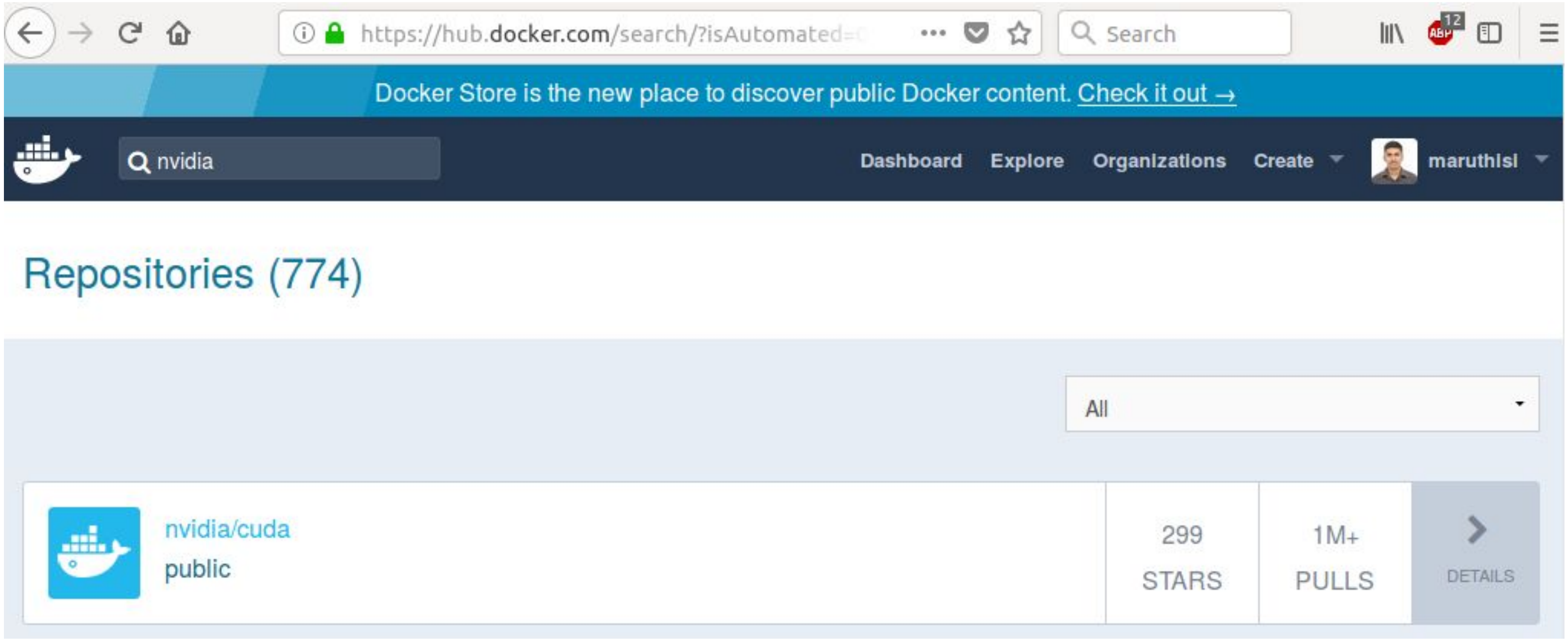
Docker components

Docker has three major components

- Docker repository/hub
- Docker image
- Docker container

Image Courtesy: Wikipedia

Docker hub


https://hub.docker.com/search/?isAutomated=

Docker Store is the new place to discover public Docker content. Check it out →

nvidia

Dashboard Explore Organizations Create maruthisi

Repositories (774)

All

nvidia/cuda	public	299 STARS	1M+ PULLS	DETAILS

Pulling images

- To pull/download docker images, use
  `docker pull <image>:<tag>`

```
$ docker pull ubuntu:latest
latest: Pulling from library/ubuntu
b234f539f7a1: Pull complete
...
d056eaf3dff4: Pull complete
7dc790b5527b: Pull complete
89bce857a556: Downloading [=========================================>        ] 463.4
MB/580.1 MB89bce857a556: Pull complete
Digest: sha256:3cdf1b5becfde8772e15dab594bc76de1cbbefd6c0f8533748854ab47e109ad1
Status: Downloaded newer image for ubuntu:latest
```

Docker container life-cycle

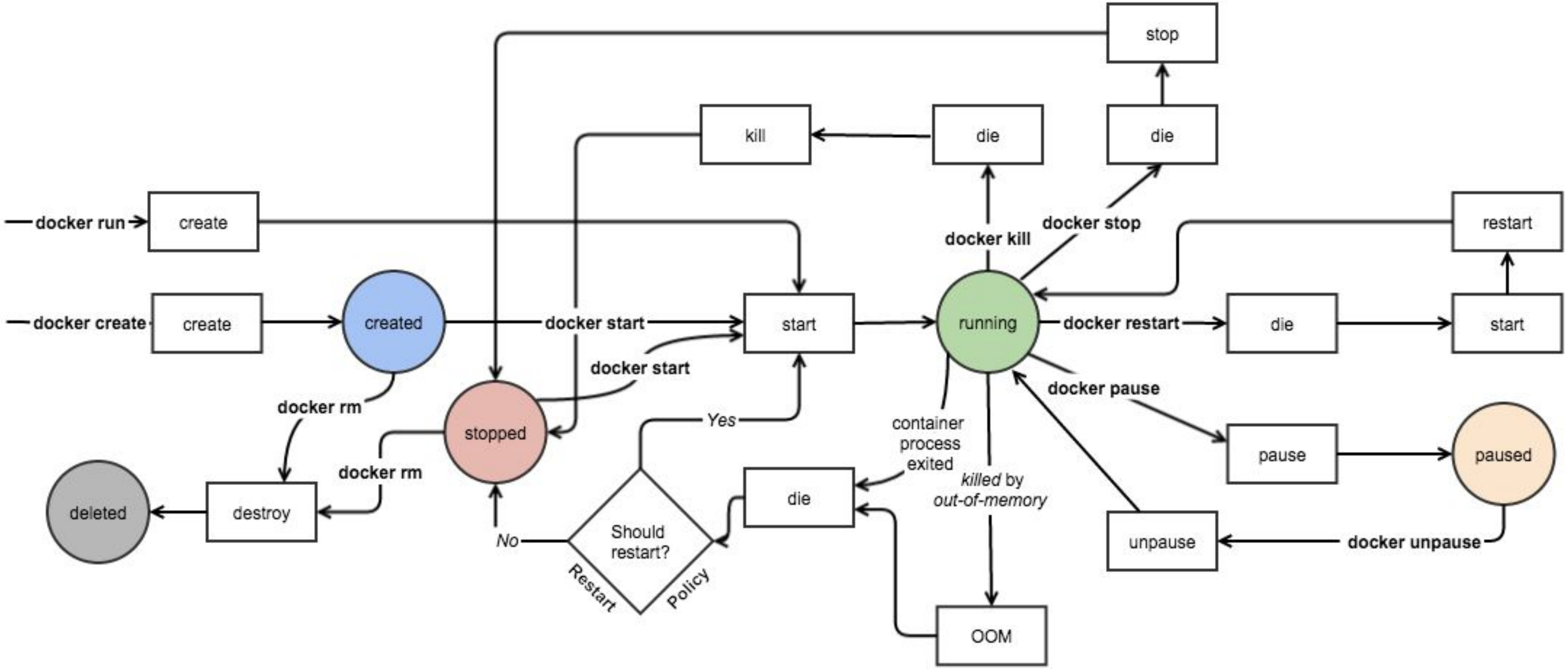


Image Courtesy: Nitin Agarwal @ medium.com

Listing images

- To list downloaded images, use
  `docker images`

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              16.04               a51debf7e1eb        2 weeks ago         116MB
ubuntu              xenial              a51debf7e1eb        2 weeks ago         116MB
ubuntu              trusty              f17b6a61de28        2 weeks ago         188MB
ubuntu              latest              93fd78260bd1        2 weeks ago         86.2MB
nvidia/cuda         9.0-base            74f5aea45cf6        3 weeks ago         134MB
centos              latest              75835a67d134        8 weeks ago         200MB
```

Running a container

- To run a container, use
  `docker run -it --name <name> <image>:<tag> <program>`

```
$  docker run -it --name centos1 centos:latest bash
[root@e7f7395af134 /]#

[root@e7f7395af134 /]# cat /etc/redhat-release
CentOS Linux release 7.5.1804 (Core)

[root@e7f7395af134 /]#
```

Listing running containers

- To list running container, use
  `docker ps`

```
$ docker ps
CONTAINER ID  IMAGE            COMMAND   CREATED          STATUS         PORTS    NAMES
e7f7395af134  centos:latest    "bash"    8 seconds ago    Up 7 seconds            centos1
```

Exiting from a container after stopping

- To stop and exit from a container, use `exit`

```
[root@e7f7395af134 /]# exit

$
```

Listing all containers

- To list all (running/exited) container, use
  `docker ps -a`

```
$ docker ps -a
CONTAINER ID  IMAGE          COMMAND   CREATED        STATUS                        PORTS   NAMES
e7f7395af134  centos:latest  "bash"    8 seconds ago   Exited (0) 5 seconds ago
centos1
```

Exiting from a container without stopping

- To exit from a container without stopping, press
  `Ctrl+p Ctrl+q`

```
[root@e7f7395af134 /]#  Ctrl+p Ctrl+q    read escape sequence

$
```

Creating a container

- To create a container, use
  `docker create --name <name> -it <image>:<tag> <program>`

```
$ docker create --name centos2 -it centos:latest bash
916dc303760db834c1aa4a9b591605ab56c91aba97bceb6fda2ca0db564f489a

$ docker ps -a
CONTAINER ID  IMAGE                  COMMAND       CREATED                 STATUS     PORTS   NAMES
e7f7395af134  centos:latest  "bash"               31 seconds ago          Created            centos1
916dc303760d  centos:latest  "bash"               About a minute ago      Created            centos2
```

Starting a container

- To start a container, use
  `docker start <name_or_id>`

```
$ docker start centos2
centos2

$ docker ps -a
CONTAINER ID  IMAGE                COMMAND      CREATED          STATUS          PORTS      NAMES
e7f7395af134  centos:latest  "bash"         4 minutes ago    Up 3 minutes
centos1
916dc303760d  centos:latest  "bash"         3 minutes ago    Up 2 minutes
centos2
```

Attaching to a running container

- To start a container, use
  `docker attach <name_or_id>`

```
$ docker attach centos2
[root@916dc303760d /]#
```

Storage

Sharing a host directory/file to container

- To share a host directory or file to a container, use
  `docker run -v <host_dir>:<guest_dir> <image>:<tag> <program>`

```
$ docker run -v /mnt:/mnt -it ubuntu:xenial bash
```

Identity and Access

Sharing host credentials with container

- To share a host credentials with a container, use

```
docker run -u `id -u`:`id -g` -v \
    /etc/passwd:/etc/passwd <image>:<tag> <program>
```

```
$ docker run -ti -u `id -u`:`id -g` --name "ubuntu-dock" -v /etc/passwd:/etc/passwd -v
/home/maruthisi:/home/maruthisi ubuntu:xenial bash
```

Networking

Listing software defined networks (SDN)

- To list SDNs
  `docker network ls`

```
$ ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:f7:e7:61:9e
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          ...
enp0s25   Link encap:Ethernet  HWaddr 90:1b:0e:e5:90:3e
          inet addr:192.168.136.108  Bcast:192.168.136.255  Mask:255.255.255.0
          ...

$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
70176b89a07a        bridge              bridge              local
a8e63013eb97        host                host                local
```

Using host network

- To launch an ubuntu docker with host networking, use
  `docker run -it --net host <image>:<tag> <command>`

```
$ docker run -it --net host  ubuntu:xenial bash
```

Using bridge network

- To launch an ubuntu docker with bridge networking, use
  `docker run -it --net bridge <image>:<tag> <command>`

```
$ docker run -it --net bridge  ubuntu:xenial bash
```

Creating custom bridge network

- To create a custom bridge network and router

```
docker network create --subnet=<network>/<prefix> <name>
```

```
$ docker network create --subnet=172.18.0.0/16 mysdn1

$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
70176b89a07a        bridge              bridge              local
a8e63013eb97        host                host                local
f85eb2d101ef        mysdn1              bridge              local
b9550c4a7b89        none                null                local
```

Using custom bridge network

- To launch an ubuntu docker with host networking, use
  `docker run -it --net <name> <image>:<tag> <command>`

```
$ docker run -it --net mysdn1 ubuntu:xenial bash
# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:ac:12:00:02
          inet addr:172.18.0.2  Bcast:172.18.255.255  Mask:255.255.0.0
          ...
```

“apt update” and “ apt install net-tools” inside the container to install ifconfig

Resource Allocation

Limiting RAM usage

- Using cgroups docker containers could be reserved/restricted resources.
- Use `--memory` to limit the memory (RAM) usage.
- Use `--memory-swap` to limit the virtual memory (RAM+Swap) usage.

```
$ docker run -it --memory=512m --memory-swap=1g ubuntu:latest bash
```

Limiting CPU usage

- Use `--cpuset-cpus` to restrict the usable CPUs.

```
$ docker run -it --cpuset-cpus=0,2 ubuntu:latest bash
```

Container security concerns

Privilege Escalation

- A container started by an unprivileged user can get root shell on the host.
  - Every container is started as a child process of the daemon running as root.
  - Child processes also run as root

On the host

```
$ cat /etc/hostname
```

Launch a guest and modify the file

```
$ docker run -it -v /etc/hostname:/etc/hostname --name ub1 ubuntu:latest bash
# id
# vi /etc/hostname
```

On the host

```
$ cat /etc/hostname
```

Lack of isolation

- A container started by an unprivileged user can be attached by another user. This is not acceptable in multi-tenant environment.

From one user

Launch a guest and modify the file

```
$ id
$ docker run -it --name ub2 ubuntu:latest bash
# top
```

From another user

```
$ id
$ docker attach ub2
```

Solutions/Workarounds

Solutions to the isolation and privilege escalation.

- Docker Enterprise Edition uses Role Based Access Control (RBAC)
  - Container daemon runs as root. A child process (called proxy daemon) is created with the same RBAC context as the user.
  - Every container is started as a child process of the proxy-daemon.
- AWS uses docker community edition, but runs containers inside VMs to provide isolation.

Workaround to the privilege escalation

- Idmapping technique is used to map host's uid/gid to non-existent uid/gid in guest.

Q & A