Docker Introduction

Tags

💡 document author: Yongjie Lyu Yongjie.L@outlook.com

What is docker?

Build, Ship and Run

Docker is the world’s leading software containerization platform

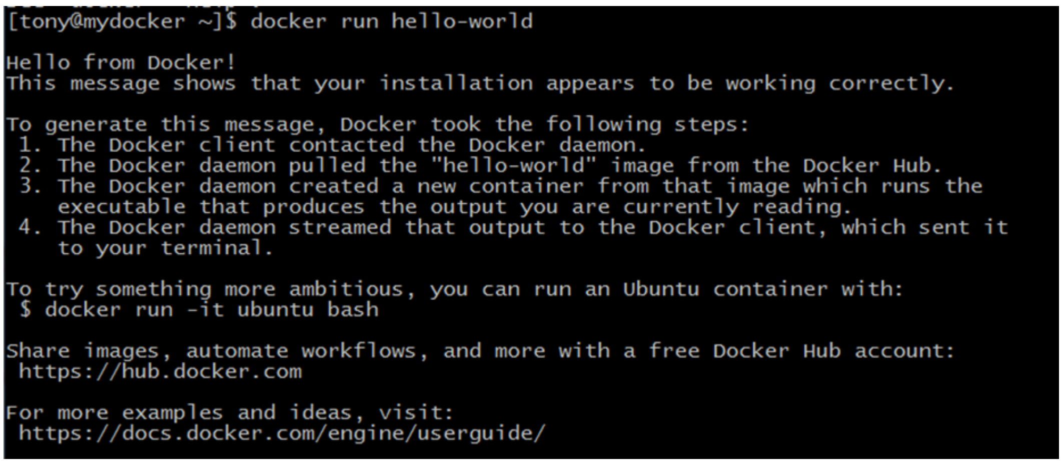
- 开源的容器引擎，可以方便的对容器进行管理
- 容器镜像，引入Docker Registry管理容器的镜像

Docker History

自从2013年年初一个叫dotCloud的PaaS服务供应商将一个内部项目Docker开源之后，这个名字在短短几年内就迅速成为一个热词。似乎一夜之间，人人都开始谈论Docker，以至于这家公司干脆出售了其所持有的PaaS平台业务，并且改名为Docker.Inc，从而专注于Docker的开发和推广。

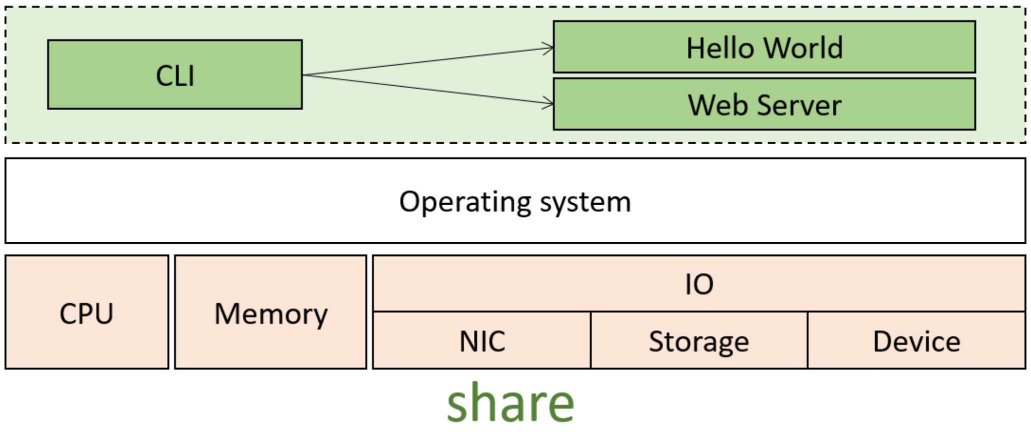
目前Docker已加入Linux基金会，遵循Apache 2.0协议

docker run hello-world

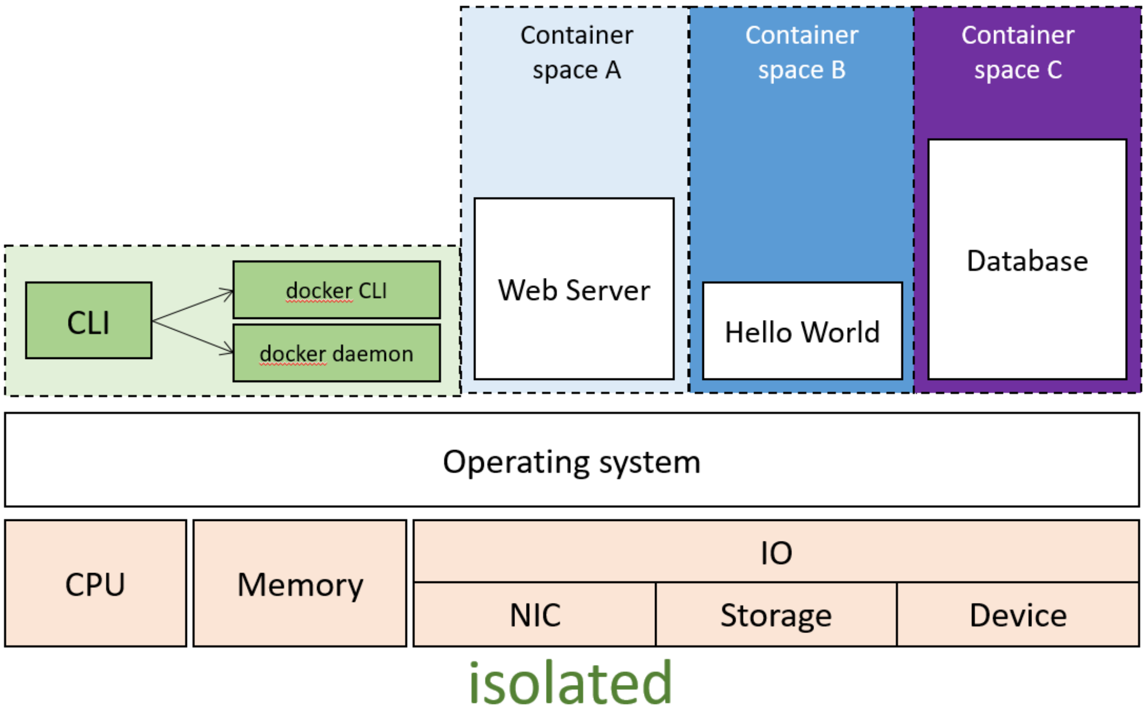


```
[tony@mydocker ~]$ docker run hello-world

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

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

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

Share images, automate workflows, and more with a free Docker Hub account:
 https://hub.docker.com

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

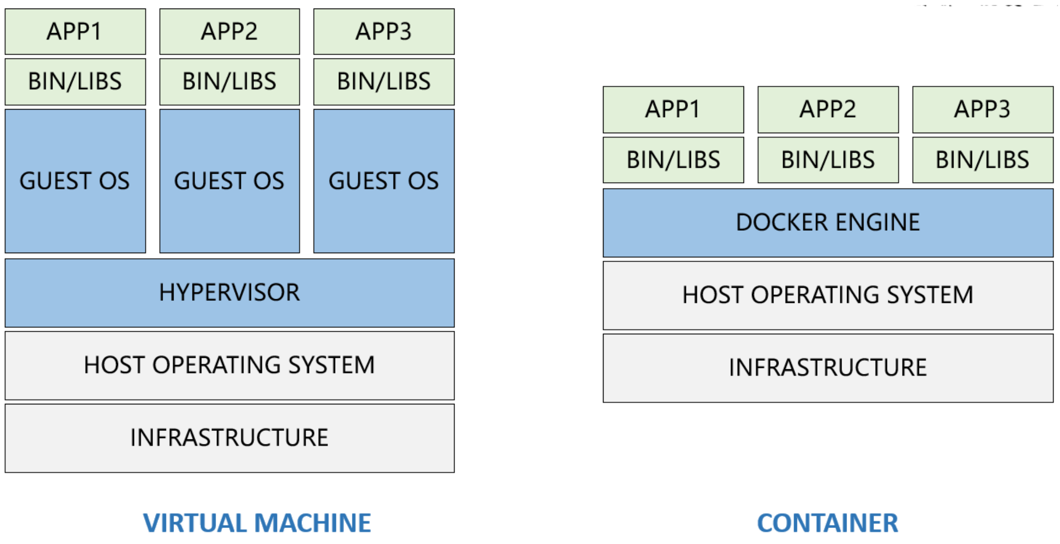
Typical application process



Docker Container



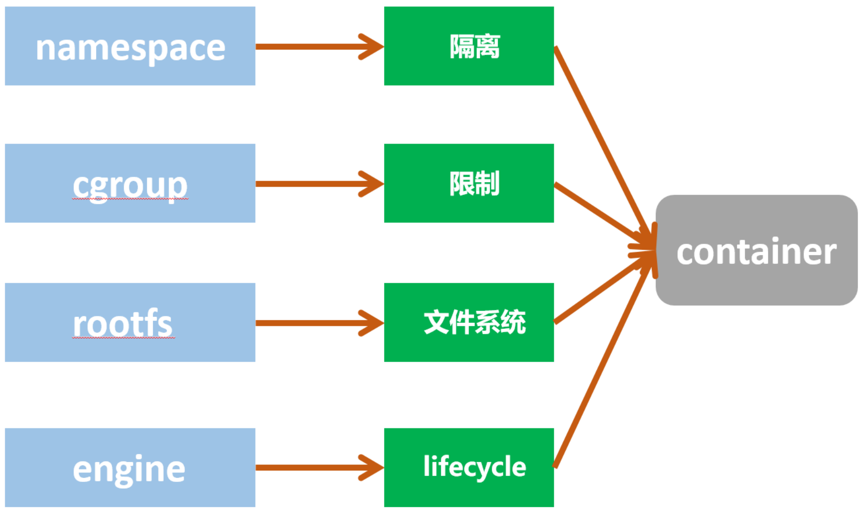
Container and Virtualization



隔离级别高　　独立的guest os　　轻量级　　启动快　　资源开销小

面向运维人员　　**面向开发人员**

Container Principle



NameSpace - 6 type

• **PID**	隔离进程ID
• **UTS**	隔离主机名和域名
• **IPC**	隔离System V IPC和POSIX消息队列
• **NET**	隔离网络资源
• **MNT**	隔离文件系统挂载点
• **USER**	隔离用户ID和组ID

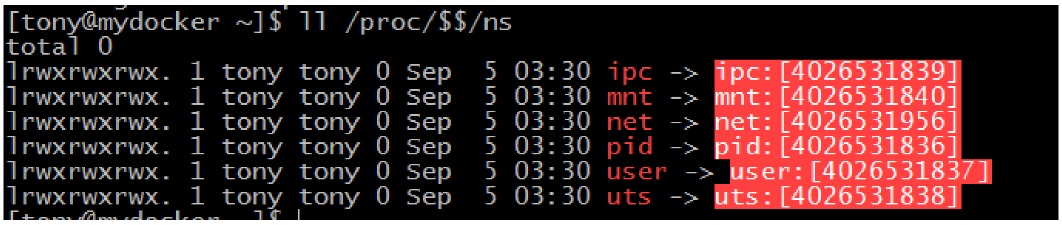
系统调用

```
int clone(int (*child_func)(void *), void *child_stack, int flags, void *arg);
int unshare(int flags);
int setns(int fd, int nstype);
```

▼ Flag

CLONE_NEWNS
CLONE_NEWIPC
CLONE_NEWNET
CLONE_NEWUTS

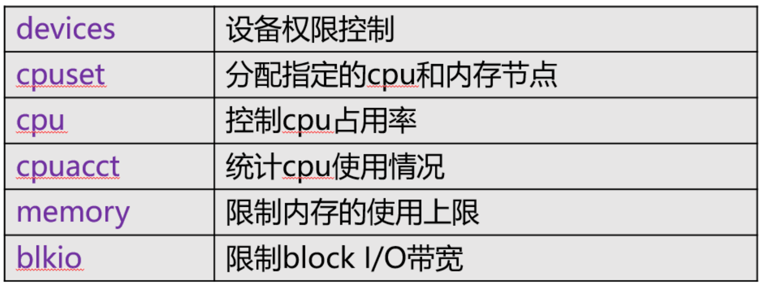
CLONE_NEWPID
CLONE_NEWUSER



```
[tony@mydocker ~]$ ll /proc/$$/ns
total 0
lrwxrwxrwx. 1 tony tony 0 Sep  5 03:30 ipc -> ipc:[4026531839]
lrwxrwxrwx. 1 tony tony 0 Sep  5 03:30 mnt -> mnt:[4026531840]
lrwxrwxrwx. 1 tony tony 0 Sep  5 03:30 net -> net:[4026531956]
lrwxrwxrwx. 1 tony tony 0 Sep  5 03:30 pid -> pid:[4026531836]
lrwxrwxrwx. 1 tony tony 0 Sep  5 03:30 user -> user:[4026531837]
lrwxrwxrwx. 1 tony tony 0 Sep  5 03:30 uts -> uts:[4026531838]
```

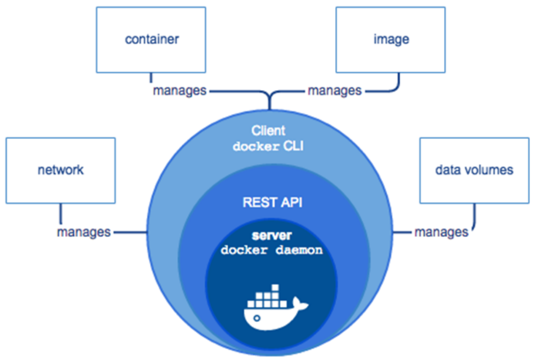
cGroup

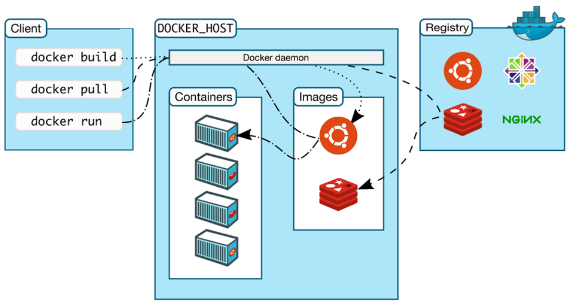
cgroup是control group的简写，用于限制和隔离一组进程对系统资源的使用，这些资源主要包括CPU、内存、block I/O和网络带宽。cgroup从2.6.24开始进入内核主线，目前各大发行版都默认打开了cgroup特性。



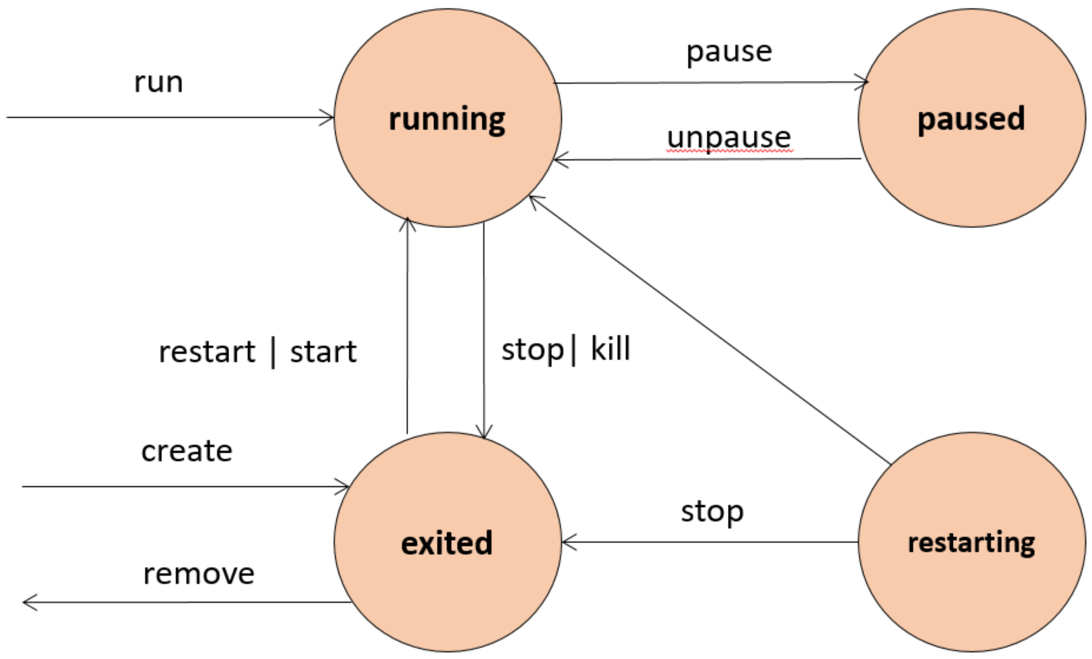
devices	设备权限控制
cpuset	分配指定的cpu和内存节点
cpu	控制cpu占用率
cpuacct	统计cpu使用情况
memory	限制内存的使用上限
blkio	限制block I/O带宽

Docker Architecture

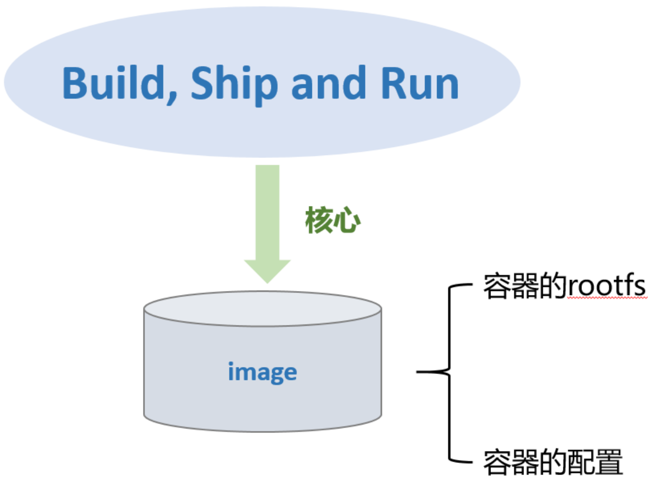




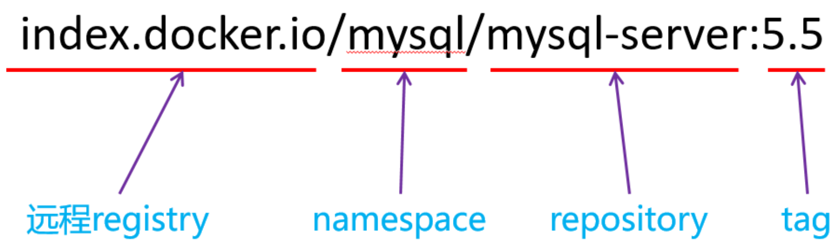
Docker container lifecycle



Docker Image

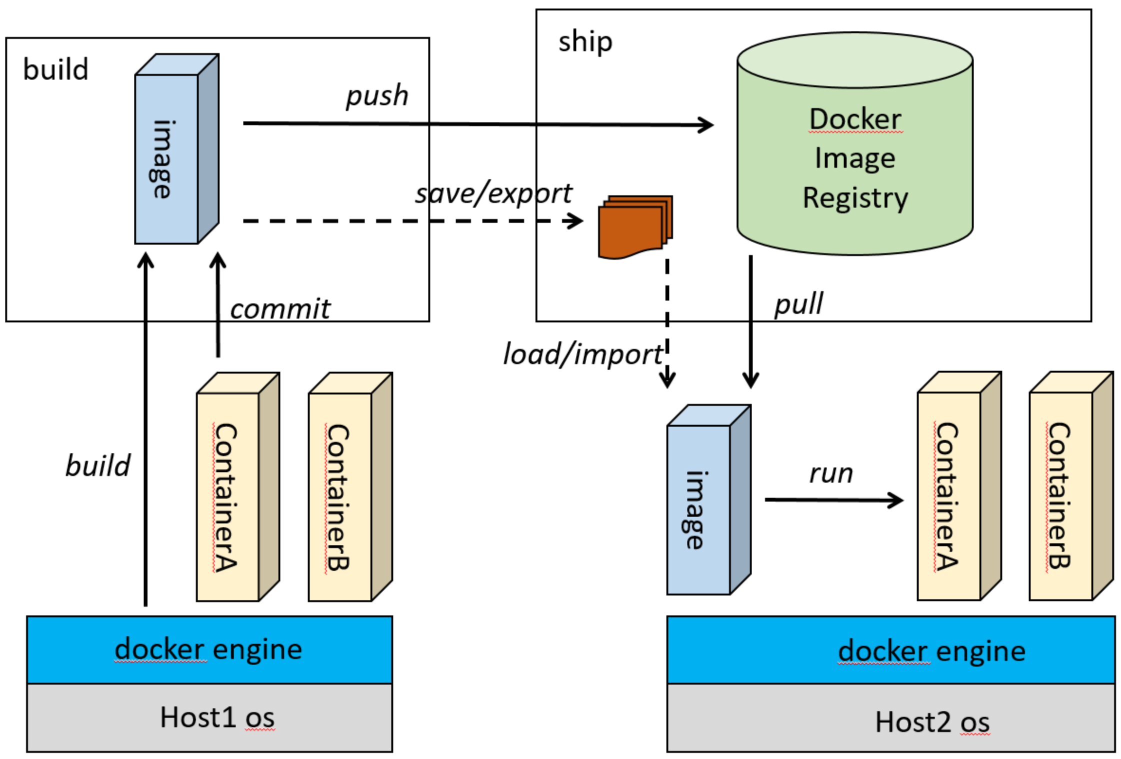


Image表现方法



```
docker run --name my-mysql \
    -e MYSQL_ROOT_PASSWORD=123456
    index.docker.io/mysql/mysql-server:5.5

docker run --name my-mysql \
    -e MYSQL_ROOT_PASSWORD=123456
    mysql/mysql-server:5.5
```

Image Container Lifecycle


build
image
push
save/export
commit
build
ContainerA
ContainerB
docker engine
Host1 os
ship
Docker Image Registry
pull
load/import
image
run
ContainerA
ContainerB
docker engine
Host2 os