## **1.1什么是docker**

Docker是一个开源的应用容器引擎，基于Go语言并遵从Apache2.0协议开源。

Docker可以让开发者打包他们的应用以及依赖包到一个轻量级、可移植的容器中，然后发布到任何流行的Linux机器上。

## **1.2docker能解决什么问题**

高效有序利用资源

在操作系统中，包括内核、文件系统、网络、PID、IPC、内存、硬盘、CPU等等，所有的资源都是应用进程直接共享的。要想实现虚拟化，除了要实现对内存、CPU、网络IO、硬盘IO、存储空间等的限制外，还要实现文件系统、网络、PID、UID、IPC等等的相互隔离。

Dokcer容器，利用Namespaces来做权限的隔离控制，利用cgroups来做资源分配，实现某些进程在彼此隔离的命名空间中运行。大家虽然都共用一个内核和某些运行时环境（例如一些系统命令和系统库），但是彼此却看不到，都以为系统中只有自己的存在。

**1.3 Docker的底层实现**

Docker底层实现使用了linux内核的一些技术

Docker底层的核心技术包括Linux上的命名空间（Namespaces）、控制组（Controlgroups）、Union文件系统（Unionfilesystems）和容器格式（Containerformat）。

**1.4 Namespces命名空间**

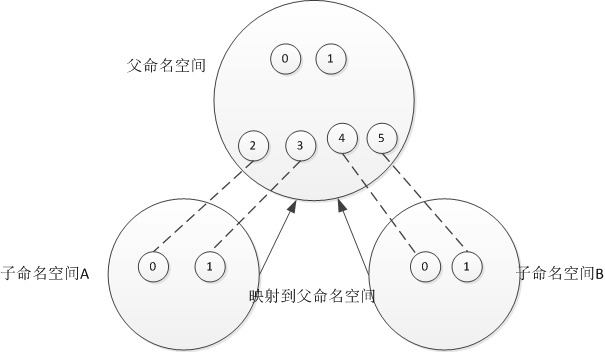
Docker使用了Namespces这一技术实现容器的隔离工作区、进程需要的资源在资源隔离的情况下，减少冲突，每一个容器都运行在独立的命名空间中，权限也会限制在这个命名空间中。

命名空间，从名字上看，应该是类似于包含许多名字的空间，打个比方，三年一班的小明和三年二班的小明，虽说他们名字是一样的，但是所在班级不一样，那么，在全年级排行榜上面，即使出现两个名字一样的小明，也会通过各自的学号来区分。对于学校来说，每个班级就相当于是一个命名空间，这个空间的名称是班级号。班级号用于描述逻辑上的学生分组信息，至于什么学生分配到1班，什么学生分配到2班，那就由学校层面来统一调度。

Linux Namespaces机制提供一种资源隔离方案。PID,IPC,Network等系统资源不再是全局性的，而是属于某个特定的Namespace。每个namespace下的资源对于其他namespace下的资源都是透明，不可见的。因此在操作系统层面上看，就会出现多个相同pid的进程。系统中可以同时存在两个进程号为0,1,2的进程，由于属于不同的namespace，所以它们之间并不冲突。而在用户层面上只能看到属于用户自己namespace下的资源，例如使用ps命令只能列出自己namespace下的进程。这样每个namespace看上去就像一个单独的Linux系统。



    命名空间建立系统的不同视图，对于每一个命名空间，从用户看起来，应该像一台单独的Linux计算机一样，有自己的init进程(PID为0)，其他进程的PID依次递增，A和B空间都有PID为0的init进程，子容器的进程映射到父容器的进程上，父容器可以知道每一个子容器的运行状态，而子容器与子容器之间是隔离的。



# **1.4.1Linux Namespace的6大类型**

1. PID Namespace

- 提供进程隔离能力

1. Net Namespace

- 提供网络隔离能力

3 IPC Namespace

- 提供进程间通信的隔离能力

1. MNT Namespace

- 提供磁盘挂载点和文件系统的隔离能力

5 UTS Namespace

- Unix分时操作系统

- 提供主机名、内核、版本隔离能力

1. User Namespace

- 提供用户隔离能力

**1.4.2 pid 命名空间：进程号命名空间，进行进程间的隔离**

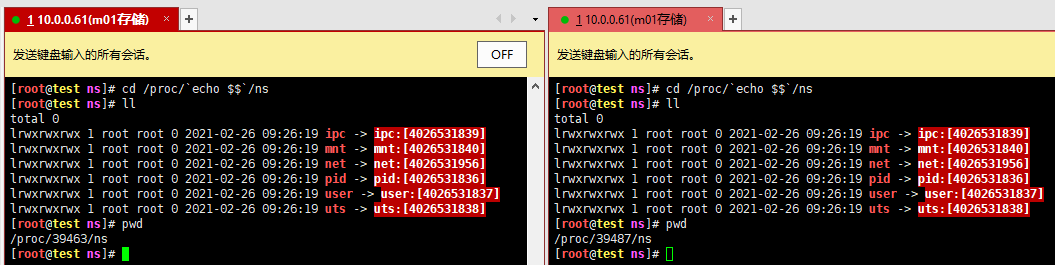
1. 不同用户的进程就是通过pid命名空间隔离开的，且不同命名空间中可以有相同pid。

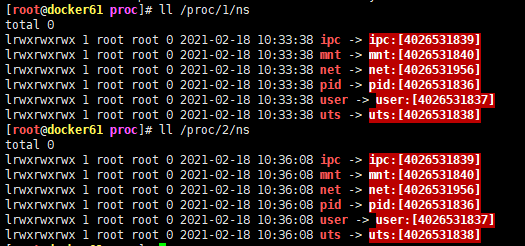
2. 每一个进程在其对应的 /proc/[pid]/ns 下都有其 namespace 信息.

- proc虚拟目录，显示内存中信息（进程、服务信息、内核）

- 目录中的数据都存储到内存中，重启系统会重置

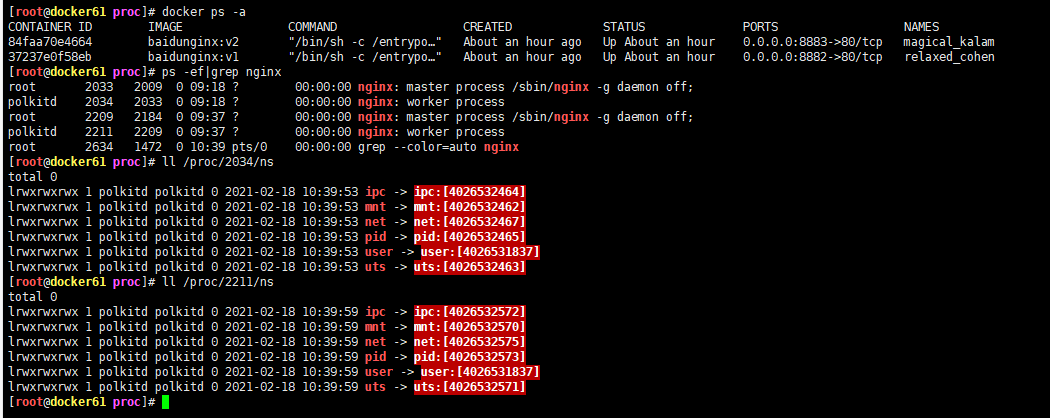
1. 查看宿主机上的进程的namespace信息





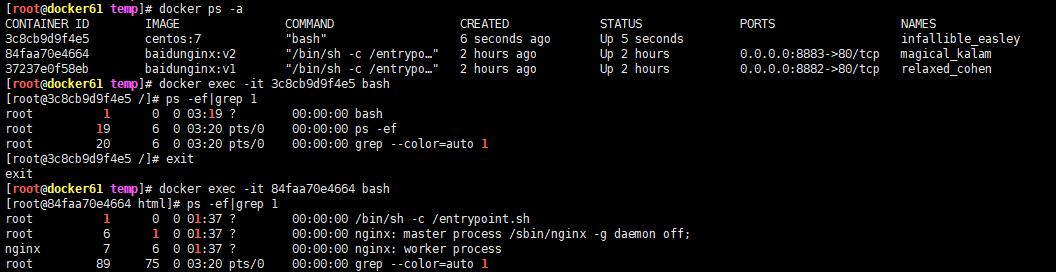
发现宿主机上面的两个进程之间的命名空间编号是一样的。这两个进程是在一个命名空间的，没有进行隔离。

1. 查看容器的进程的namespace信息



发现容器的命名空间和宿主机及其他容器的命名空间是不同的，是隔离的。

1. 查询不同容器间的进程

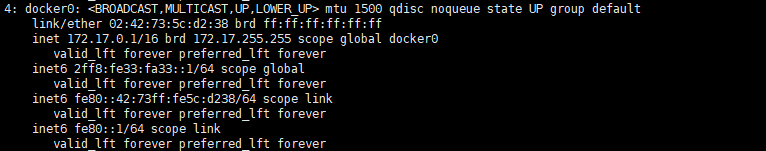


**1.4.2 net 命名空间：网络命令空间，管理网络接口**

1. 有了pid命名空间,每个命名空间中的pid能够相互隔离，但是网络端口还是共享host的端口。网络隔离是通过net命名空间实现的，每个net命名空间有独立的网络设备,IP地址,路由表,/proc/net目录。这样每个容器的网络就能隔离开来。

1. Docker网络如何通信：

Docker默认采用veth的方式，将容器中的虚拟网卡同host上的一个Docker网桥docker0连接在一起

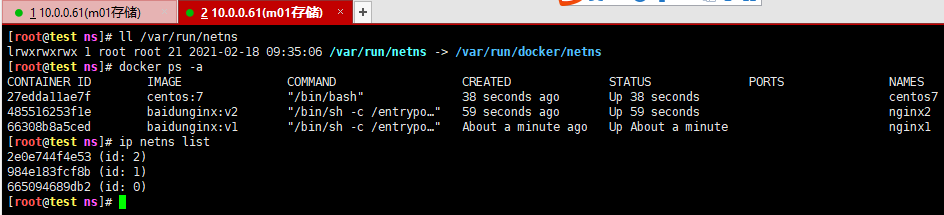


* 1. 显示所有的虚拟网络命名空间：

- ip netns list

- 默认查询/var/run/netns目录

-链接docker netns到系统默认目录：ln -s /var/run/docker/netns /var/run/netns



* 1. 查看链接到网桥上的虚拟网口信息：brctl show docker0

- yum install -y bridge-utils

每个容器都会分配一个网桥上的虚拟网口，并将docker0的IP地址设置为默认网关，容器发起的网络流量通过宿主机的iptables规则进行转发

[root@test ns]# brctl show docker0

bridge name bridge id STP enabled interfaces

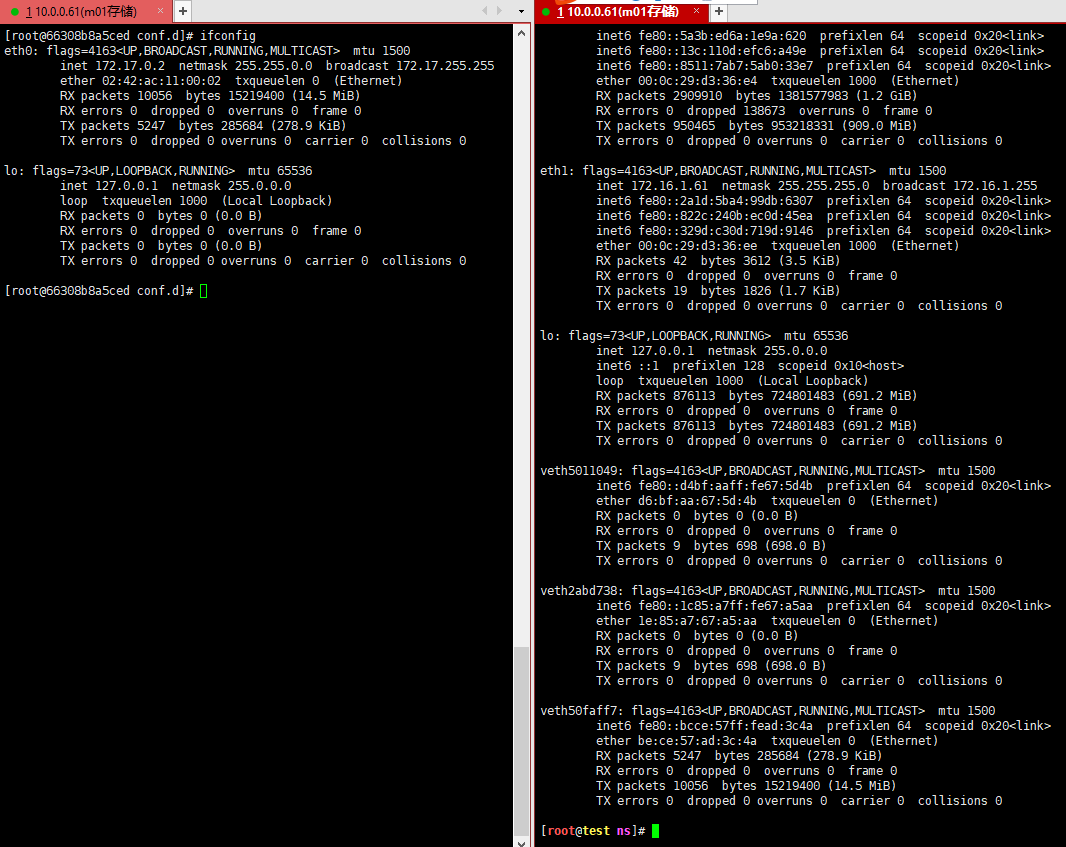
docker0 8000.02422bfb6585 no veth2abd738

veth5011049

veth50faff7

* 1. Linux虚拟网络技术

1. 宿主机和容器各创建一个虚拟接口veth
2. 宿主机虚拟接口连接到docker0网桥或指定网桥
3. 容器中的虚拟网口修改名为eth0
4. 从网桥中分配一个闲置的IP地址给容器eth0，并设置网关为docker0的IP网关

 查看容器网络详细信息：

[root@docker61 ~]# docker inspect 84faa70e4664



[root@docker61 ~]# docker exec -it 84faa70e4664 bash

[root@84faa70e4664 html]# ping 172.17.0.1

PING 172.17.0.1 (172.17.0.1) 56(84) bytes of data.

64 bytes from 172.17.0.1: icmp\_seq=1 ttl=64 time=0.129 ms

[root@84faa70e4664 html]# ping 172.17.0.2

PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.

64 bytes from 172.17.0.2: icmp\_seq=1 ttl=64 time=0.067 ms

1. Docker网络模式：

none：无网络模式

bridge：默认模式，相当于NAT

host：共用宿主机Network NameSapce

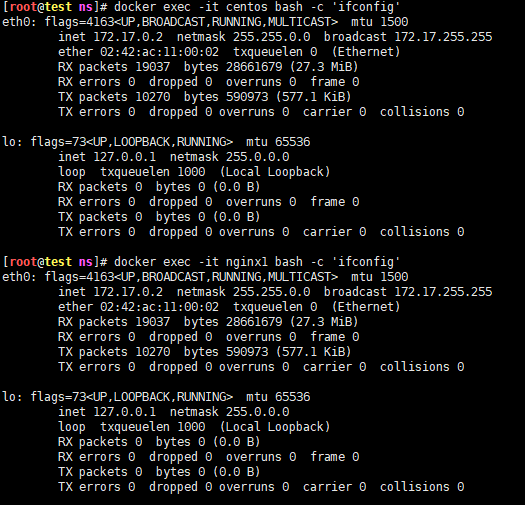
container：与其他容器公用Network Namespace

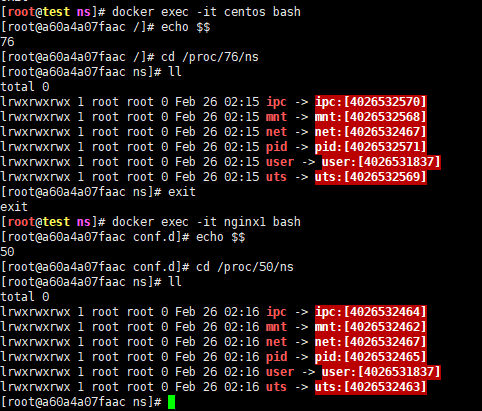
如何共用net命名空间:

docker run -di --name nginx1 b56267eccd67 bash

docker run -di --name centos --net=container:nginx1 centos:7

注意共用net命名空间后，端口可能出现冲突情况，例如两台nginx用的都是80端口，那么其中第二台nginx不能启动





**1.4.3 mnt 命名空间：管理文件系统，挂载**

类似chroot，将一个进程放到一个特定的目录执行。mnt命名空间允许不同命名空间的进程看到的文件结构不同，这样每个命名空间中的进程所看到的文件目录就被隔离开了。同chroot不同，每个命名空间中的容器在/proc/mounts的信息只包含所在命名空间的mount point。

未隔离的情况：

[root@docker61 test]# echo $$

3728

[root@docker61 test]# ls

a1 a2 a3

[root@docker61 test]# echo $$

3889

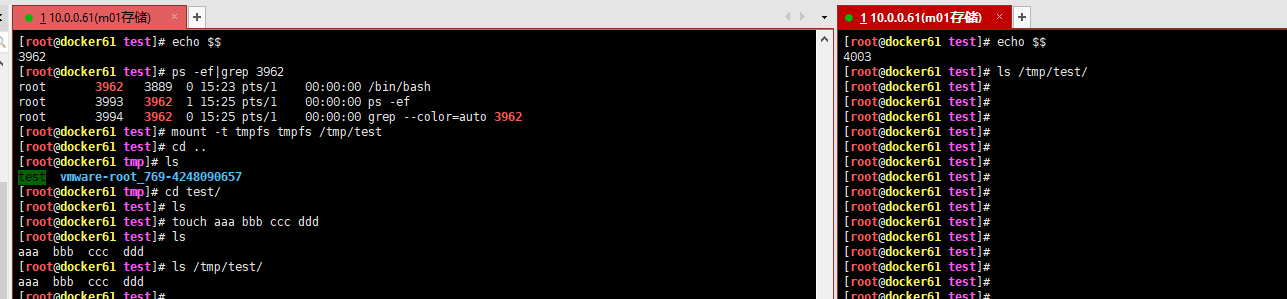
[root@docker61 test]# ls

a1 a2 a3

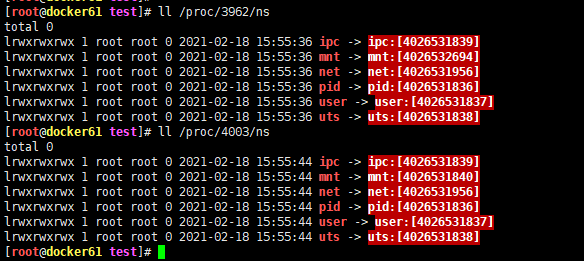
有隔离的情况：

1. *unshare --mount /bin/bash*重新启动一个bash进程，其中的mnt namespace是隔离的。
2. tmpfs格式挂载：

tmpfs是一种虚拟内存文件系统,正如这个定义它最大的特点就是它的存储空间在VM里面，VM是由linux内核里面的vm子系统管理的东西，现在大多数操作系统都采用了虚拟内存管理机制。linux下面VM(virtual memory）的大小由RM(Real Memory)和swap组成。

****

1. 查看mnt命令空间：

****

**1.4.4 uts 命名空间：Unix分时系统，可以隔离内核、版本、主机名等**

UTS("UNIXTime-sharingSystem")命名空间允许每个容器拥有独立的host name和domain name,使其在网络上可以被视作一个独立的节点而非主机上的一个进程。

[root@docker61 test]# unshare --uts /bin/bash

[root@docker61 test]# echo $$

4132

[root@docker61 test]# hostname

docker61

**[root@docker61 test]# hostname**

**docker61**

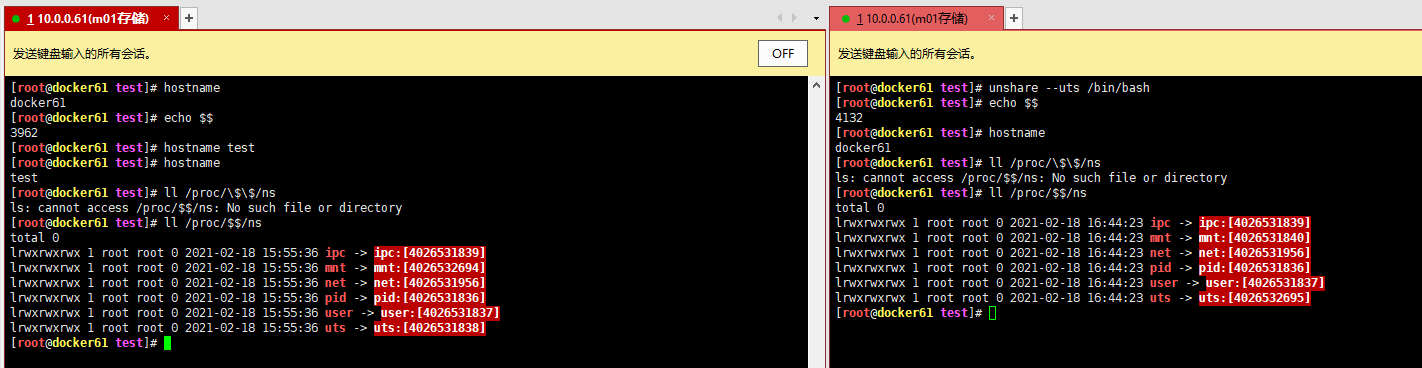
**[root@docker61 test]# echo $$**

**3962**

**[root@docker61 test]# hostname test**

**[root@docker61 test]# hostname**

**test**

****

**1.4.5 Ipc 命名空间：管理进程间通信资源（消息隔离）**

容器中进程交互采用了Linux常见的进程间交互方法,包括信号量、消息队列和共享内存等。然而同VM不同的是，容器的进程间交互实际上还是host上具有相同pid命名空间中的进程间交互，因此需要在IPC资源申请时加入命名空间信息，每个IPC资源有一个唯一的32位id。