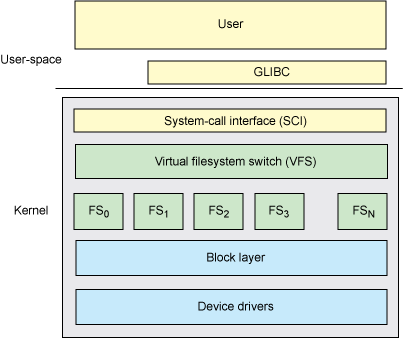
**概述**

用户的权限管理至关重要，尤其对生产环境服务器来说，因为root的用户的权力太大（破坏性大），大多数生产环境应用为了防止被网络上的各种“坏人”通过应用漏洞拿到root权限，一般会使用普通用户权限来运行应用。但同时，权限管理又往往是一个系统中最复杂的部分。稍有不慎，应用无法启动都是最简单的问题，更多运行过程中程序崩溃、报错，导致莫名其妙的问题产生却又无从下手排查。

Docker和宿主机共用一个内核，所以只能通过namespace隔离的方式来实现用户及其权限的隔离，使得实现的复杂度又上了一个台阶，以至于早期内核实现了user namespace功能但却一直很不稳定，而docker也一直默认不开启user namespace功能。

本文从操作系统用户权限管理入手，介绍Docker用户权限和操作系统用户权限如何进行映射，如何进行隔离以及Docker中user namespace一些基本的机制与原理。本文不涉及Windows版本及其Docker。本篇主要介绍基于root启动Docker Daemon的情况，Rootless Mode会在下一篇当中介绍。

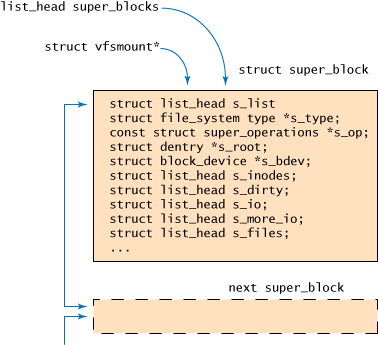
**Linux的权限机制**



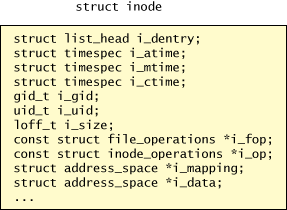
1-1 Linux文件系统架构

Linux内核采用了虚拟文件系统VFS，用户态操作文件的时候，只需要通过系统调用去进行read、write、lseek等文件操作操作，并不需要关心具体是什么文件系统，各个文件系统自己去对接VFS。

与Windows不同的是，当文件系统挂载到Linux系统上去的时候，VFS会在内核中为该文件系统生成一个超级块结构，超级块结构用于存储文件系统的信息，如文件类型、大小、inode信息等。而inode中会存储文件的权限、属主、属组等信息。



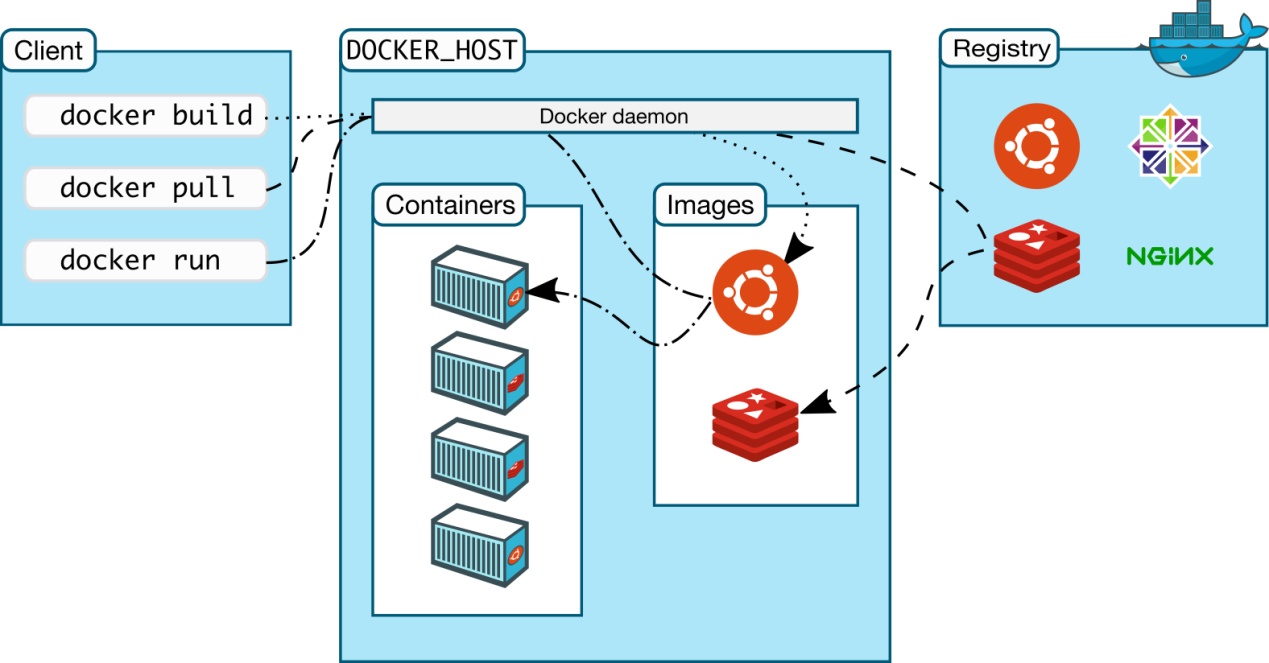
1-2 super\_blocks结构



1-3 struct inode结构

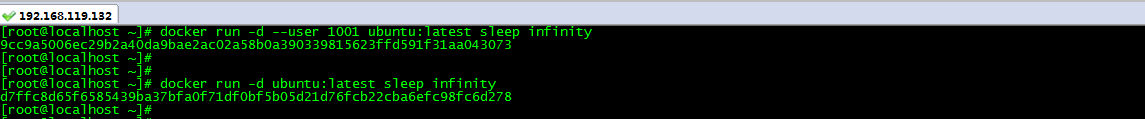
Linux在启动进程的时候，除了记录进程的pid、ppid等信息，还会记录启动进程的用户信息。当进程对文件系统中的文件进行操作时，内核会通过操作系统会将用户信息去获取用户的uid和pid，然后与inode中的文件权限信息进行比较，只有符合权限要求的操作请求才会被通过。

**Docker默认用户机制**



1-3 docker架构

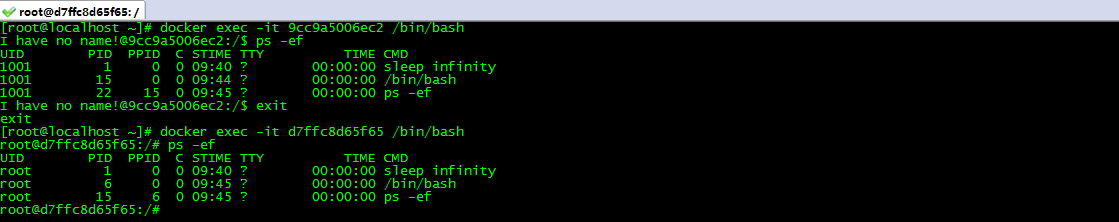
Docker采用了CS架构，通过图中我们可以看到，Docker client发送请求信息，而由Docker daemon处理请求，完成Docker相关操作。

在非Rootless Mode（systemctl start docker）时候，默认情况下，Docker daemon会以指定用户权限启动进程。如下图，我们分别指定用户和不指定用户来启动一个容器：

1-4 指定用户创建容器

我们来看看两个进程的属主：

1-5 宿主机中看容器进程

1001即wangcf的用户id。可以看出，在不指定用户的情况下，docker容器以root用户启动应用进程。在指定用户的情况下，docker会以指定用户权限的身份运行应用。下面我们看看docker容器里面的进程信息：

1-6 容器中看应用进程

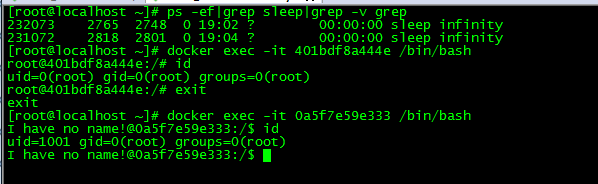
可以看到，容器内和容器外是相同的用户，那么这就存在两个问题：一个是安全问题，如果未指定用户，docker容器进程会以root权限启动。另一个是如果容器内想使用root用户那就必须使用宿主机的root。

为了解决这个问题，docker提供了userns-remap配置项来调用内核（3.8以上版本）的username space来进行用户映射。如图所示：



1-7 docker用户映射配置参数

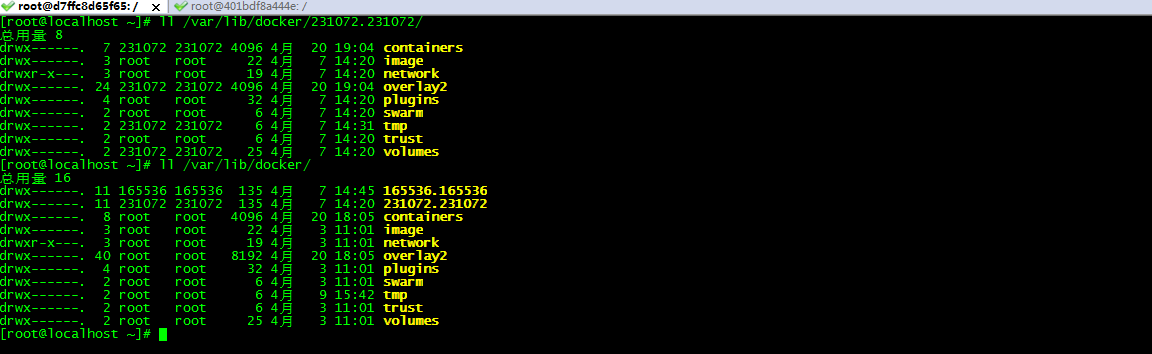
在增加配置项并重启docker后，docker会默认创建一个名为dockermap的用户来作为映射用户，username space提供了子用户的概念，当创建docker容器时候，会将docker进程用户映射为该用户的子用户：



1-8 映射后的进程用户id

图中，依然创建了两个容器，一个未指定用户，一个指定了1001用户。但是无论有没有指定用户，我们可以看出来，docker都将其映射成了非系统用户，232073和231072分别是dockermap的两个子用户。子用户的概念可以详细去了解username space，此处不做深入讨论。这样就解决了创建容器默认会使用root用户的问题，同时，即使容器内使用root用户，也会给映射成dockermap的子用户，从而解决了容器的安全性隐患。

当然，我们由于使用了映射用户，那么创建docker容器时指定挂载目录或者容器时候必须要给足够的目录权限，否则可能导致容器出错而进程退出，如图，我们可看到，容器创建时候，docker数据目录进行了重新映射，由于映射空间的切换，映射前的docker镜像和容器都将会读取不到。



1-9 映射数据目录了权限

**后记**

Docker使用内核提供的username space技术解决docker在容器创建时候的用户问题。但是由于docker daemon是以root用户启动的，当docker daemon开启tcp端口接受远程调用时候可能被人利用。原来docker发生的渗透漏洞，因为docker daemon是以root用户启动，然后开启了监听端口，被恶意用户获取到了操作系统的root权限。下一篇我们将讨论Docker Daemon的Rootless（non-root user）模式。

参考资料:

<http://www.grabsun.com/article/2012/871996.html>

<https://medium.com/@mccode/understanding-how-uid-and-gid-work-in-docker-containers-c37a01d01cf>

<https://docs.docker.com/engine/security/userns-remap/>

<https://success.docker.com/article/introduction-to-user-namespaces-in-docker-engine>