Docker

b站ID 佩雨小神仙

公众号 快乐小神仙

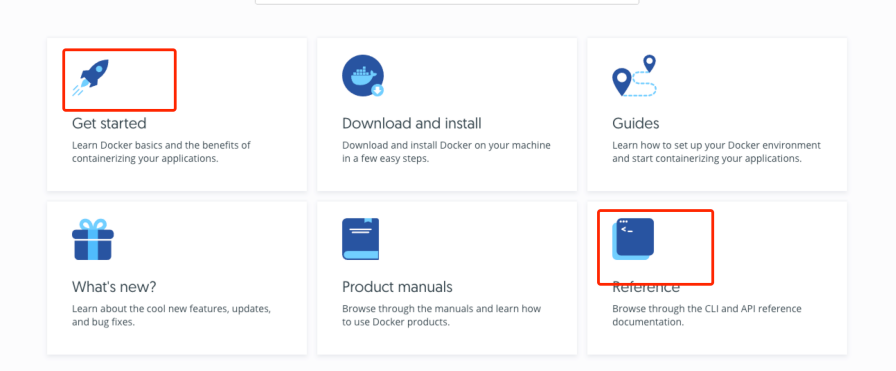
喜欢的话记得点个关注呦~

小神仙寄语：

首先你需要有一定的Docker基础

学习东西一定要去官网学习，不能用第二手信息取代官网。

官网https://docs.docker.com/



The underlying technology

Docker is written in the Go programming language and takes advantage of several features of the Linux kernel to deliver its functionality.

Docker使用了`linux内核`的一些技术来实现。

Namespaces 命名空间

Docker uses a technology called `namespaces` to provide the isolated workspace called the *container*. When you run a container, Docker creates a set of *namespaces* for that container.

These namespaces provide a layer of isolation. Each aspect of a container runs in a separate namespace and its access is limited to that namespace.

Docker Engine uses namespaces such as the following on Linux:

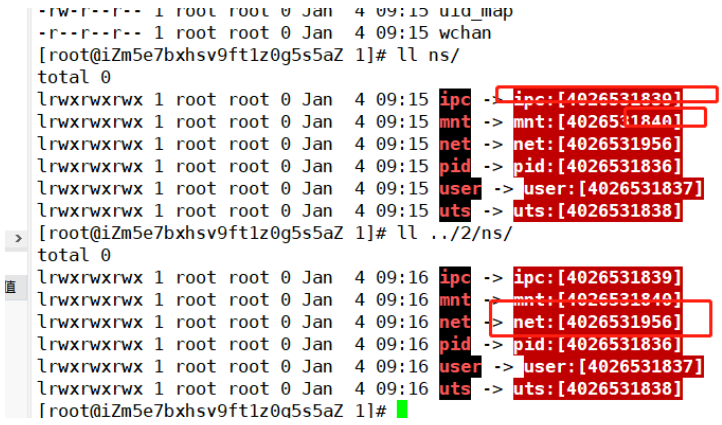
- **The `pid` namespace:** Process isolation (PID: Process ID).
- **The `net` namespace:** Managing network interfaces (NET: Networking).
- **The `ipc` namespace:** Managing access to IPC resources (IPC: InterProcess Communication).
- **The `mnt` namespace:** Managing filesystem mount points (MNT: Mount).
- **The `uts` namespace:** Isolating kernel and version identifiers. (UTS: Unix Timesharing System).

实验：

首先 cd /pro

查看进程为1 的命名空间。

查看进程为2的命名空间。

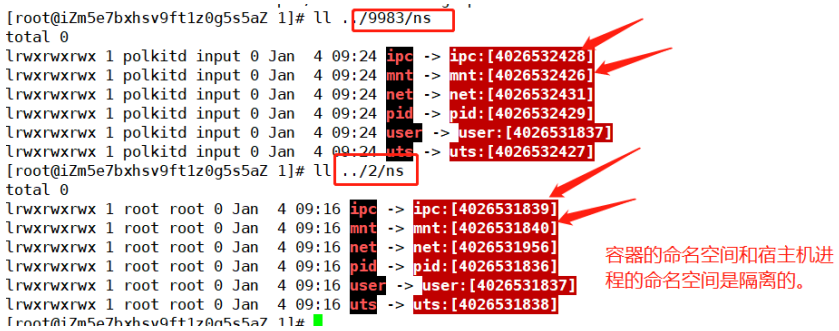


发现宿主机上面的两个进程之间的命名空间的编号是一样的。

这说明两个进程是在一个命名空间的。

创建一个redis的容器

```
docker run -d redis
ps -ef | grep redis // 查找到容器的进程号
```



发现是隔离的命名空间。

man手册使用

当你不会什么东西的时候，Ask a man

man 1 2 3 4 5 6 7 8

你熟悉的事情，其实你根本就不知道。

学习 知之为知之 不知为不知。

```
1   Executable programs or shell commands
2   System calls (functions provided by the kernel)
3   Library calls (functions within program libraries)
4   Special files (usually found in /dev)
5   File formats and conventions eg /etc/passwd
6   Games
7   Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
8   System administration commands (usually only for root)
9   Kernel routines [Non standard]
```

Linux 的線上手冊分為幾個主要的章節（sections），每個章節對應不同的類別：

- 1：可執行的程式或是 shell 指令。
- 2：系統呼叫（system calls，Linux 核心所提供的函數）。
- 3：一般函式庫函數。

- 4：特殊檔案（通常位於 `/dev`）。
- 5：檔案格式與協定，如 `/etc/passwd`
- 6：遊戲。
- 7：雜項（巨集等，如 `man(7)`、`groff(7)`）。
- 8：系統管理者指令（通常是管理者 `root` 專用的）。
- 9：Kernel routines（非標準）。

Control groups

Docker Engine on Linux also relies on another technology called *control groups* (`cgroups`). A cgroup limits an application to a specific set of resources.

Control groups allow Docker Engine to share available hardware resources to containers and optionally enforce limits and constraints. For example, you can limit the memory available to a specific container.

实验：

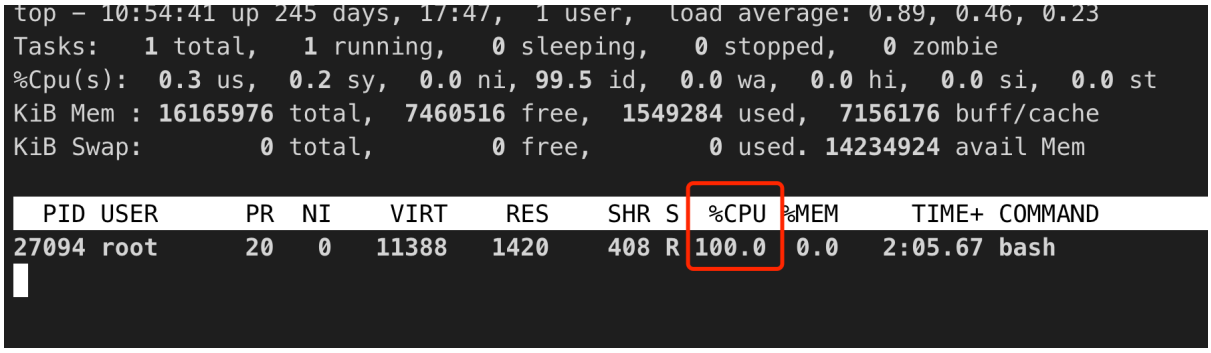
```
//下载工具
yum search libcgroup
yum install libcgroup-tools.x86_64

//创建控制组
cgcreate -g cpu:echo_test
ls /sys/fs/cgroup/cpu/echo_test

//写一个循环
while :; do :; done&

//查看当前的进程的资源负载
top -p 27094
```

可以发现该进程占满



设置一下这个group的cpu利用的限制

```
echo 30000 > /sys/fs/cgroup/cpu/echo_test/cpu.cfs_quota_us
```

把这个进程加入cgroup

```
echo 27094 >> /sys/fs/cgroup/cpu/echo_test/tasks
```

然后，就会在top中看到CPU的利用立马下降成30%了。（前面我们设置的30000就是20%的意思

Union file systems 联合文件系统

Union file systems, or UnionFS, are file systems that operate by creating layers, making them very lightweight and fast. Docker Engine uses UnionFS to provide the building blocks for containers. Docker Engine can use multiple UnionFS variants, including AUFS, btrfs, vfs, and DeviceMapper.

文件系统是什么?

计算机的文件系统是一种存储和组织计算机数据的方法，它使得对其访问和查找变得容易，文件系统使用文件和树形目录的抽象逻辑概念代替了硬盘和光盘等物理设备使用数据块的概念，用户使用文件系统来保存数据不必关心数据实际保存在硬盘（或者光盘）的地址为多少的数据块上，只需要记住这个文件的所属目录和文件名。在写入新数据之前，用户不必关心硬盘上的那个块地址没有被使用，硬盘上的存储空间管理（分配和释放）功能由文件系统自动完成，用户只需要记住数据被写入到了哪个文件中。

Container format

Docker Engine combines the namespaces, control groups, and UnionFS into a wrapper called a container format. The default container format is `libcontainer`. In the future, Docker may support other container formats by integrating with technologies such as BSD Jails or Solaris Zones.

docker 容器就是宿主机上面的一个进程

通过配置linux namespace隔离进程的运行环境

通过cgroups来限制进程的资源

以下为中文版本~

依赖的底层技术

1、名字空间

名字空间是 Linux 内核一个强大的特性。每个容器都有自己单独的名字空间，运行在其中的应用都像是在独立的操作系统中运行一样。名字空间保证了容器之间彼此互不影响。

2.1 pid 名字空间

不同用户的进程就是通过 pid 名字空间隔离开的，且不同名字空间中可以有相同 pid。所有的 LXC 进程在 Docker 中的父进程为Docker进程，每个 LXC 进程具有不同的名字空间。同时由于允许嵌套，因此可以很方便的实现嵌套的 Docker 容器。

2.2 net 名字空间

有了 pid 名字空间, 每个名字空间中的 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 名字空间实现的， 每个 net 名字空间有独立的 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。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。

2.3 ipc 名字空间

容器中进程交互还是采用了 Linux 常见的进程间交互方法(interprocess communication – IPC), 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 名字空间中的进程间交互，因此需要在 IPC 资源申请时加入名字空间信息，每个 IPC 资源有一个唯一的 32 位 id。

2.4 mnt 名字空间

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

2.5 uts 名字空间

UTS(“UNIX Time-sharing System”) 名字空间允许每个容器拥有独立的 hostname 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。

2.6 user 名字空间

每个容器可以有不同的用户和组 id, 也就是说可以在容器内用容器内部的用户执行程序而非主机上的用户。

3、控制组

控制组（cgroups）是 Linux 内核的一个特性，主要用来对共享资源进行隔离、限制、审计等。只有能控制分配到容器的资源，才能避免当多个容器同时运行时的对系统资源的竞争。

控制组技术最早是由 Google 的程序员 2006 年起提出，Linux 内核自 2.6.24 开始支持。

控制组可以提供对容器的内存、CPU、磁盘 IO 等资源的限制和审计管理。

4、联合文件系统

联合文件系统（UnionFS）是一种分层、轻量级并且高性能的文件系统，它支持对文件系统的修改作为一次提交来一层层的叠加，同时可以将不同目录挂载到同一个虚拟文件系统下(unite several directories into a single virtual filesystem)。

联合文件系统是 Docker 镜像的基础。镜像可以通过分层来进行继承，基于基础镜像（没有父镜像），可以制作各种具体的应用镜像。

另外，不同 Docker 容器就可以共享一些基础的文件系统层，同时再加上自己独有的改动层，大大提高了存储的效率。

Docker 中使用的 AUFS（AnotherUnionFS）就是一种联合文件系统。 AUFS 支持为每一个成员目录（类似 Git 的分支）设定只读（readonly）、读写（readwrite）和写出（whiteout-able）权限, 同时 AUFS 里有一个类似分层的概念, 对只读权限的分支可以逻辑上进行增量地修改(不影响只读部分的)。

Docker 目前支持的联合文件系统种类包括 AUFS, btrfs, vfs 和 DeviceMapper。

5、容器格式

最初，Docker 采用了 LXC 中的容器格式。自 1.20 版本开始，Docker 也开始支持新的 libcontainer 格式，并作为默认选项。

对更多容器格式的支持，还在进一步的发展中。