



湖南大学
HUNAN UNIVERSITY

沙箱技术、容器技术与 Docker

王任杰



湖南大学
HUNAN UNIVERSITY

第一部分 沙箱技术



沙箱技术是一种隔离环境技术，它为程序提供一个受限的运行环境，使得程序在其中的操作不会对系统的其他部分造成影响。

就像在一个封闭的 “箱子” 里运行程序，限制其对系统资源的访问权限。

在 Ubuntu 系统中，沙箱技术可以通过多种方式实现，例如利用 Linux 的命名空间（Namespace）和控制组（Cgroup）技术。

命名空间用于隔离进程、文件系统、网络等资源，使得沙箱中的进程无法看到或访问沙箱外的资源。

Cgroup 则用于限制进程对 CPU、内存等资源的使用量，防止沙箱中的程序过度占用系统资源。



沙箱技术

如何在 Linux 中安装 Firejail

如图所示，可以通过使用 git 命令从项目的 github 页面下载最新的包来完成安装。

```
$ git clone
```

```
https://github.com/netblue30/firejail.git
```

```
cd firejail
```

```
./configure && make && sudo make  
install-strip
```

在安装中可能会报错，需要根据报错信息进行配置



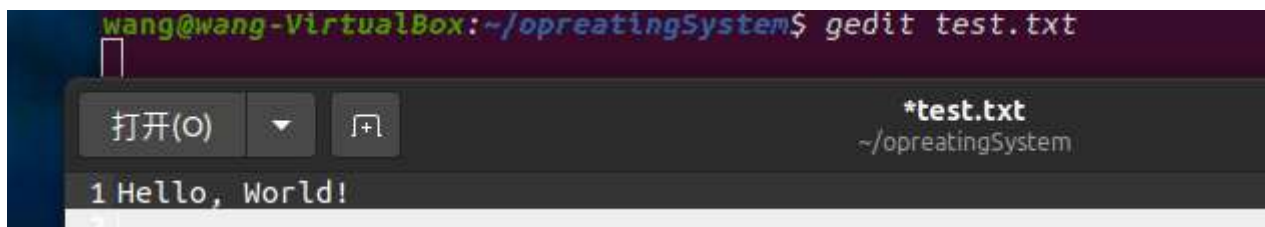


沙箱技术

创建一个test文本用于测试



```
wang@wang-VirtualBox: ~$ echo "Hello, World!" > test.txt
```

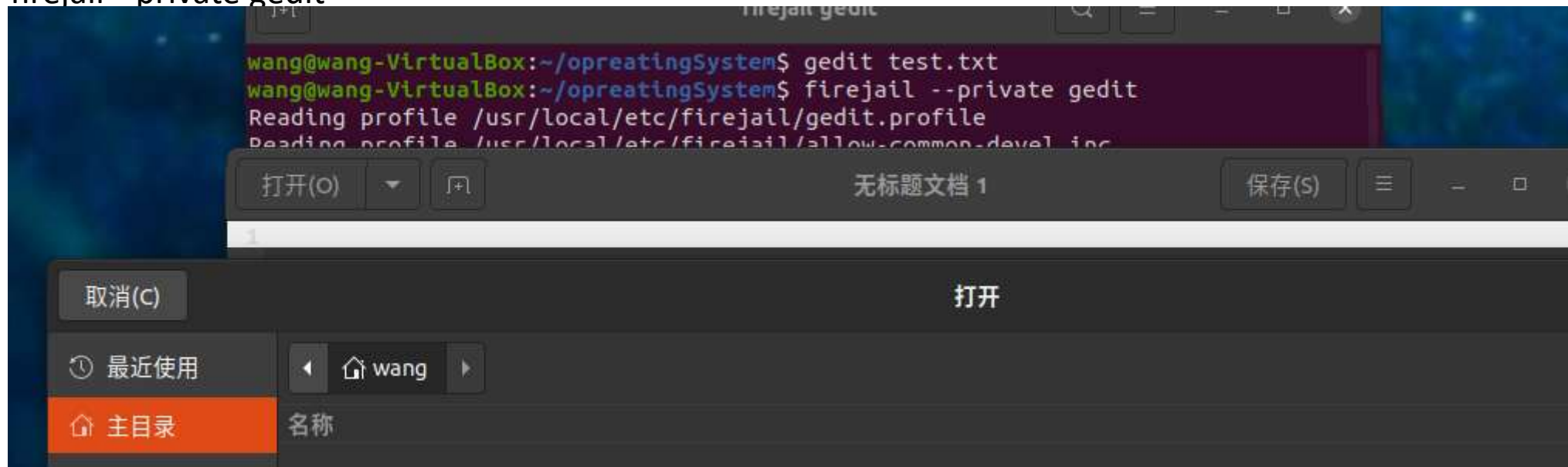


```
wang@wang-VirtualBox:~/opreatingSystem$ gedit test.txt
```

1 Hello, World!

在一个隔离的、私有的文件系统环境中启动 gedit 文本编辑器。gedit 在运行过程中只能访问这个临时文件系统，无法直接访问宿主系统的其他文件和目录

firejail --private gedit



```
wang@wang-VirtualBox:~/opreatingSystem$ gedit test.txt
wang@wang-VirtualBox:~/opreatingSystem$ firejail --private gedit
Reading profile /usr/local/etc/firejail/gedit.profile
Reading profile /usr/local/etc/firejail/allow-common-devel.inc
```

无标题文档 1

取消(C) 打开

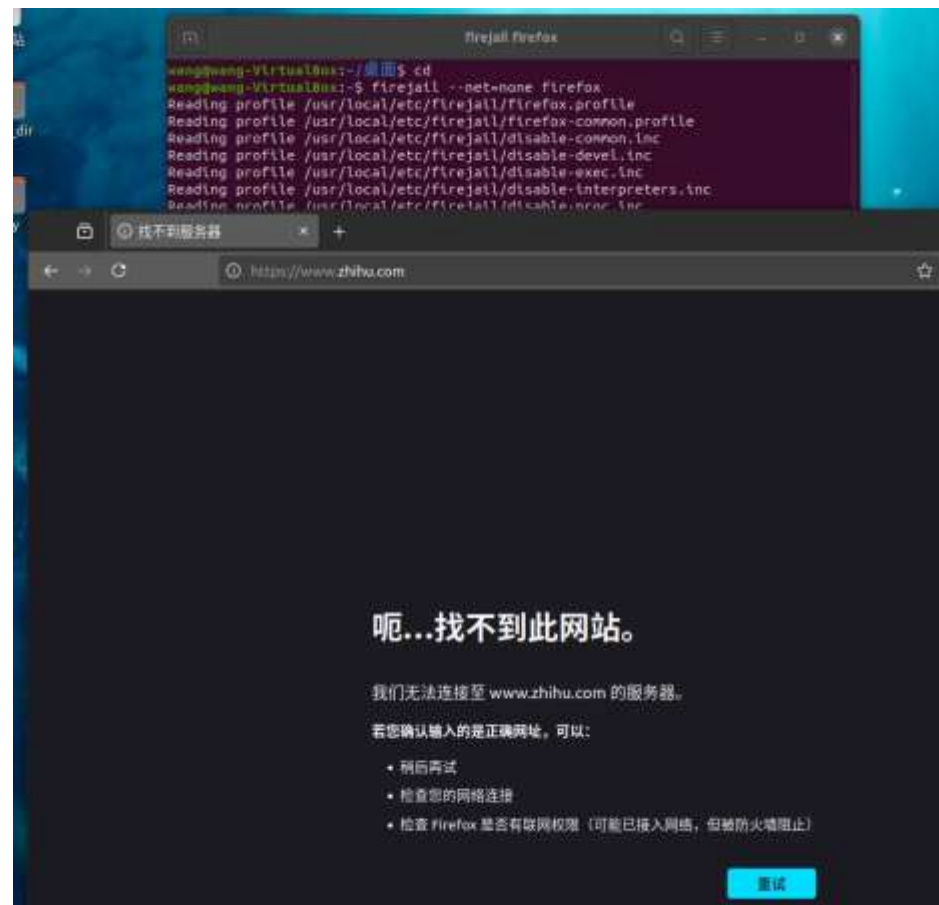
最近使用

主目录

wang

名称

firejail --net=none firefox





```
wang@wang-VirtualBox: ~  
wang@wang-VirtualBox:~$ stress --cpu 4 --vm 1 --vm-bytes 1G  
stress: info: [2942] dispatching hogs: 4 cpu, 0 io, 1 vm, 0 hdd  
  
1 [|||||] 100.0% Tasks: 125, 276 thr; 4 running  
2 [|||||] 100.0% Load average: 2.10 0.72 0.27  
3 [|||||] 100.0% Uptime: 00:03:01  
4 [|||||] 100.0%  
Mem[|||||] 798M/3.82G  
Swp[|||||] 0K/1.14G  
  
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command  
2946 wang 20 0 3864 100 0 R 87.6 0.0 0:18.71 stress --cpu 4  
2944 wang 20 0 1027M 33040 208 R 84.3 0.8 0:19.09 stress --cpu 4  
2947 wang 20 0 3864 100 0 R 78.9 0.0 0:19.60 stress --cpu 4  
2945 wang 20 0 3864 100 0 R 78.3 0.0 0:19.69 stress --cpu 4  
2943 wang 20 0 3864 100 0 R 74.2 0.0 0:19.15 stress --cpu 4
```

stress --cpu 4 --vm 1 --vm-bytes 1G

```
firejail stress --cpu 4 --vm 1 --vm-bytes 1G  
wang@wang-VirtualBox:~$ firejail --cpu=0 --rlimit-as=256m stress --cpu 4 --vm 1  
--vm-bytes 1G  
Reading profile /usr/local/etc/firejail/default.profile  
Reading profile /usr/local/etc/firejail/disable-common.inc  
Reading profile /usr/local/etc/firejail/disable-programs.inc  
Reading profile /usr/local/etc/firejail/landlock-common.inc  
  
wang@wang-VirtualBox: ~  
1 [|||||] 100.0% Tasks: 124, 271 thr; 4 running  
2 [|||||] 0.0% Load average: 2.25 0.82 0.43  
3 [|||||] 0.0% Uptime: 00:11:21  
4 [|||||] 0.0%  
Mem[|||||] 816M/3.82G  
Swp[|||||] 0K/1.14G  
  
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command  
3047 wang 20 0 4708 3044 2564 S 0.0 0.1 0:00.04 firejail --cpu=0 --rlimit-  
621 messagebu 20 0 9756 6176 3924 S 0.0 0.2 0:00.87 /usr/bin/dbus-daemon --sys
```

firejail --cpu=0 --rlimit-as=256m stress --cpu 4 --vm 1 --vm-bytes 1G



湖南大学
HUNAN UNIVERSITY

第二部分 容器技术



容器技术是一种轻量级的虚拟化技术，它将应用程序及其依赖环境打包成一个容器，使得应用程序可以在不同的环境中运行，并且相互隔离。与传统的虚拟机相比，容器更加轻量级，启动速度更快，资源占用更少。

Ubuntu 基于 Linux 内核，容器技术主要依赖于 Linux 的 Namespace 和 Cgroup 技术，与沙箱技术有相似之处，但容器更侧重于应用程序的打包和部署。每个容器都有自己独立的文件系统、进程空间和网络配置，容器之间相互隔离，互不影响。



安装 LXC

`sudo apt-get install lxc`

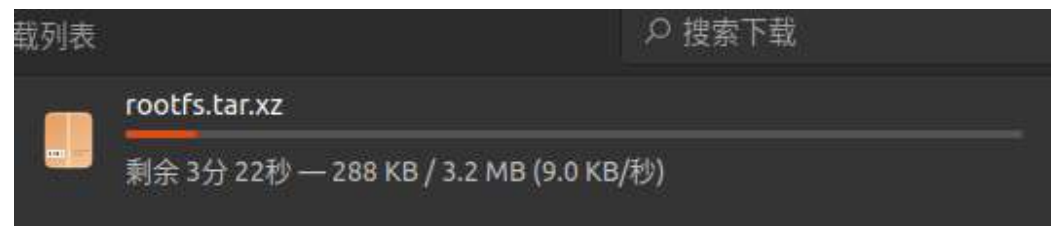
以下命令检查 Linux 内核是否具有必需的配置

```
wang@wang-VirtualBox: ~  
wang@wang-VirtualBox:~$ lxc-checkconfig  
LXC version 4.0.12  
Kernel configuration not found at /proc/config.gz; searching...  
Kernel configuration found at /boot/config-5.15.0-136-generic  
--- Namespaces ---  
Namespaces: enabled  
Utsname namespace: enabled  
Ipc namespace: enabled  
Pid namespace: enabled  
User namespace: enabled  
Network namespace: enabled  
--- Control groups ---  
Control groups: enabled
```

创建一个特权容器

`lxc-create --name mycontainer --template download -- --dist alpine --release 3.19 --arch amd64`

lxc会利用 download 模板从远程镜像源下载 Alpine Linux 3.19 版本的 amd64 架构镜像，并基于此镜像创建名为 mycontainer 的容器





湖南大学
HUNAN UNIVERSITY

第三部分 Docker

Docker 是一种基于容器技术的开源平台，它简化了容器的创建、部署和管理过程。Docker 提供了一套完整的工具链，包括 Docker 引擎、Docker 镜像、Docker 仓库等，使得开发者可以更轻松地使用容器技术。

Docker 可以看作是容器技术的一种高级应用

```
wang@wang-VirtualBox:~$ systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset:
   Active: active (running) since Tue 2025-04-22 17:46:01 CST; 1min 7s ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 8370 (dockerd)
       Tasks: 10
      Memory: 21.1M
         CGroup: /system.slice/docker.service
                └─8370 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/cont>

4月 22 17:46:01 wang-VirtualBox dockerd[8370]: time="2025-04-22T17:46:01.346116>
4月 22 17:46:01 wang-VirtualBox dockerd[8370]: time="2025-04-22T17:46:01.374528>
4月 22 17:46:01 wang-VirtualBox dockerd[8370]: time="2025-04-22T17:46:01.473756>
4月 22 17:46:01 wang-VirtualBox dockerd[8370]: time="2025-04-22T17:46:01.684338>
4月 22 17:46:01 wang-VirtualBox dockerd[8370]: time="2025-04-22T17:46:01.711321>
4月 22 17:46:01 wang-VirtualBox dockerd[8370]: time="2025-04-22T17:46:01.711474>
4月 22 17:46:01 wang-VirtualBox dockerd[8370]: time="2025-04-22T17:46:01.740796>
4月 22 17:46:01 wang-VirtualBox dockerd[8370]: time="2025-04-22T17:46:01.763398>
4月 22 17:46:01 wang-VirtualBox systemd[1]: Started Docker Application Containe>
4月 22 17:46:01 wang-VirtualBox dockerd[8370]: time="2025-04-22T17:46:01.764287>
lines 1-21/21 (END)
```



Docker 安装过程的一些问题

```
wang@wang-VirtualBox:~$ sudo docker pull hello-world
Using default tag: latest
Error response from daemon: Get "https://registry-1.docker.io/v2/": net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)
```

```
wang@wang-VirtualBox:~$ sudo mkdir -p /etc/docker
wang@wang-VirtualBox:~$ sudo tee /etc/docker/daemon.json <<- 'EOF'
> {
>   "registry-mirrors": [
>     "https://do.nark.eu.org",
>     "https://dc.j8.work",
>     "https://docker.m.daocloud.io",
>     "https://dockerproxy.com",
>     "https://docker.mirrors.ustc.edu.cn",
>     "https://docker.nju.edu.cn"
>   ]
> }
> EOF
{
  "registry-mirrors": [
    "https://do.nark.eu.org",
    "https://dc.j8.work",
    "https://docker.m.daocloud.io",
    "https://dockerproxy.com",
    "https://docker.mirrors.ustc.edu.cn",
    "https://docker.nju.edu.cn"
  ]
}
wang@wang-VirtualBox:~$ sudo systemctl daemon-reload
wang@wang-VirtualBox:~$ sudo systemctl restart docker
```

```
wang@wang-VirtualBox:~$ sudo 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. (amd64)
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 ID:
<https://hub.docker.com/>

For more examples and ideas, visit:
<https://docs.docker.com/get-started/>



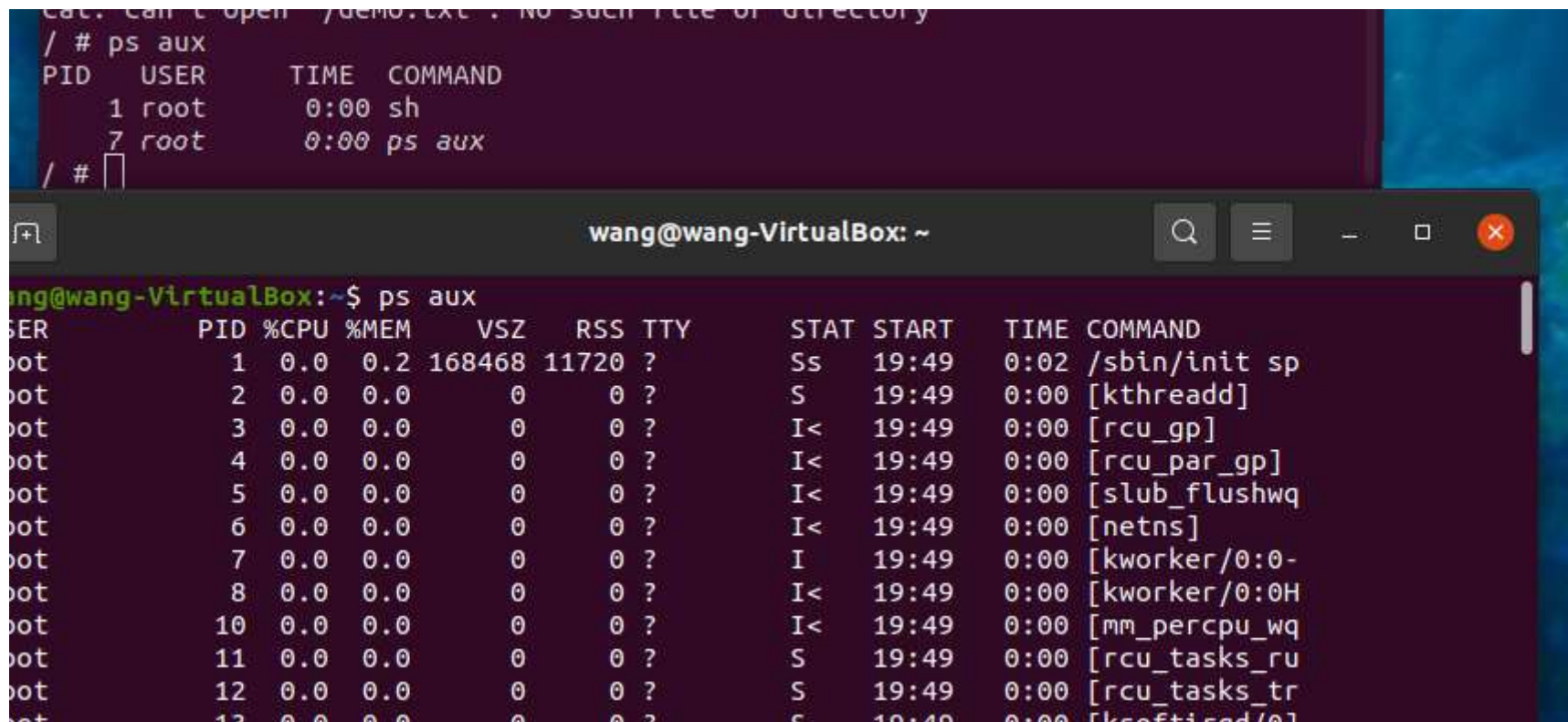
Docker

容器内的文件系统是隔离的，销毁容器后修改丢失

```
wang@wang-VirtualBox: ~  
wang@wang-VirtualBox:~$ service docker start  
wang@wang-VirtualBox:~$ docker run -it --rm alpine sh  
docker: permission denied while trying to connect to the Docker daemon socket at  
unix:///var/run/docker.sock: Head "http://%2Fvar%2Frun%2Fdocker.sock/_ping": di  
al unix /var/run/docker.sock: connect: permission denied  
  
Run 'docker run --help' for more information  
wang@wang-VirtualBox:~$ sudo docker run -it --rm alpine sh  
[sudo] wang 的密码:  
Unable to find image 'alpine:latest' locally  
latest: Pulling from library/alpine  
f18232174bc9: Pull complete  
Digest: sha256:a8560b36e8b8210634f77d9f7f9efd7ffa463e380b75e2e74aff4511df3ef88c  
Status: Downloaded newer image for alpine:latest  
/ # echo "Hello from Container!" > /demo.txt  
/ # cat /demo.txt  
Hello from Container!  
/ # exit  
wang@wang-VirtualBox:~$
```

```
wang@wang-VirtualBox:~$ sudo docker run -it --rm alpine sh  
[sudo] wang 的密码:  
/ # cat /demo.txt  
cat: can't open '/demo.txt': No such file or directory  
/ #
```

容器内只能看到自己的进程，与宿主机隔离



The screenshot shows two terminal windows. The top window is a container's shell, and the bottom window is the host's VirtualBox terminal.

Container Terminal (Top):

```
cat: can't open '/demo.txt': No such file or directory
/ # ps aux
PID  USER    TIME  COMMAND
   1  root      0:00  sh
   7  root      0:00  ps aux
/ #
```

Host Terminal (Bottom):

```
wang@wang-VirtualBox: ~$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.2	168468	11720	?	Ss	19:49	0:02	/sbin/init sp
root	2	0.0	0.0	0	0	?	S	19:49	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	19:49	0:00	[rcu_gp]
root	4	0.0	0.0	0	0	?	I<	19:49	0:00	[rcu_par_gp]
root	5	0.0	0.0	0	0	?	I<	19:49	0:00	[slub_flushwq]
root	6	0.0	0.0	0	0	?	I<	19:49	0:00	[netns]
root	7	0.0	0.0	0	0	?	I	19:49	0:00	[kworker/0:0-
root	8	0.0	0.0	0	0	?	I<	19:49	0:00	[kworker/0:0H
root	10	0.0	0.0	0	0	?	I<	19:49	0:00	[mm_percpu_wq]
root	11	0.0	0.0	0	0	?	S	19:49	0:00	[rcu_tasks_ru
root	12	0.0	0.0	0	0	?	S	19:49	0:00	[rcu_tasks_tr
root	13	0.0	0.0	0	0	?	S	19:49	0:00	[ksoftirqd/0]



Docker 的一些特点

Docker 提出了镜像的概念，将应用及其依赖项打包成一个标准化的文件。你能够从 Docker Hub 等镜像仓库拉取现成的镜像，从而避免手动配置复杂的环境，相较于前面使用LXC更加方便

```
wang@wang-VirtualBox:~$ sudo docker history alpine
[sudo] wang 的密码:
IMAGE          CREATED          CREATED BY
SIZE          COMMENT
aded1e1a5b37   2 months ago    CMD ["/bin/sh"]
0B            buildkit.dockerfile.v0
<missing>      2 months ago    ADD alpine-minirootfs-3.21.3-x86_64.tar
.gz /...       7.83MB          buildkit.dockerfile.v0
wang@wang-VirtualBox:~$
```

Docker 镜像由多个 只读层（Layer）堆叠而成，每一层代表镜像构建过程中的一个步骤（例如安装软件、复制文件、配置环境等）。这些层按顺序叠加，最终形成一个完整的文件系统视图



Docker 的一些特点

在物理存储中，两个镜像都基于 ubuntu 基础镜像，这个基础镜像对应的层只会保存一份，镜像 C 和镜像 D 会共享这一层。只有在各自安装 Python 和 Java 时产生的层才是不同的，会分别保存

容器可看作是镜像的运行实例——基于镜像创建容器时，会在镜像的只读层之上叠加一个可写层，用于存储运行时的临时数据（如修改的文件、日志、进程状态等）

```
wang@wang-VirtualBox:~$ service docker start
wang@wang-VirtualBox:~$ sudo docker run -it -v my-vol:/app/data alpine sh
/ # echo "This is a test file." > /tmp/test.txt
/ # exit
wang@wang-VirtualBox:~$ sudo docker ps -a
CONTAINER ID   IMAGE     PORTS      COMMAND      CREATED      STATUS
7b64c95eb5f6   alpine    "sh"       About a minute ago   Exited (0) 14
seconds ago   mystifying_jackson
0a35f4024a5e   alpine    "sh"       2 minutes ago       Exited (127) A
bout a minute ago   dazzling_robinson
9a51fa487dfd   alpine    "sh"       10 minutes ago      Exited (0) 9 m
inutes ago     relaxed_elgamal
af5591161a5b   hello-world  "/hello"   28 hours ago        Exited (0) 28
hours ago     kind_hypatia
wang@wang-VirtualBox:~$ docker inspect 7b64c95eb5f6 | grep "Mounts" -A 10
permission denied while trying to connect to the Docker daemon socket at un
ix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.49/con
tainers/7b64c95eb5f6/json": dial unix /var/run/docker.sock: connect: permis
sion denied
wang@wang-VirtualBox:~$ sudo docker inspect 7b64c95eb5f6 | grep "Mounts" -A
10
    "Mounts": [
      {
        "Type": "volume",
        "Name": "my-vol",
        "Source": "/var/lib/docker/volumes/my-vol/_data",
        "Destination": "/app/data",
        "Driver": "local",
        "Mode": "z",
        "RW": true,
        "Propagation": ""
      }
    ]
wang@wang-VirtualBox:~$ sudo docker inspect 9a51fa487dfd | grep "Mounts" -A
10
    "Mounts": [],
    "Config": {
      "Hostname": "9a51fa487dfd",
      "Domainname": "",
      "User": "",
      "AttachStdin": true,
      "AttachStdout": true,
      "AttachStderr": true,
      "Tty": true,
      "OpenStdin": true,
      "StdinOnce": true,
    }
wang@wang-VirtualBox:~$
```



技术	隔离级别	资源占用	应用场景
沙箱技术	进程级隔离	较低	运行不可信程序，限制程序访问权限
容器技术	应用级隔离	中等	打包和部署应用程序，实现环境一致性
Docker	应用级隔离（基于容器技术）	中等	简化容器管理，支持微服务架构



湖南大学
HUNAN UNIVERSITY

END

Thanks for watching
