# Namespace的作用

参考《趣谈Linux操作系统》的第57课：Namespace技术：内部创业公司应该独立运营。

容器虚拟化，共享操作系统内核。为了隔离不同类型的资源，保证各容器之间互不影响，Linux内核里面实现了以下几种不同类型的namespace。

* UTS，对应的宏为 CLONE\_NEWUTS，表示不同的 namespace 可以配置不同的 hostname。
* User，对应的宏为 CLONE\_NEWUSER，表示不同的 namespace 可以配置不同的用户和组。
* Mount，对应的宏为 CLONE\_NEWNS，表示不同的 namespace 的文件系统挂载点是隔离的
* PID，对应的宏为 CLONE\_NEWPID，表示不同的 namespace 有完全独立的 pid，也即一个 namespace 的进程和另一个 namespace 的进程，pid 可以是一样的，但是代表不同的进程。
* Network，对应的宏为 CLONE\_NEWNET，表示不同的 namespace 有独立的网络协议栈。

# 安装Docker

安装方法参考：

<https://developer.aliyun.com/mirror/docker-ce?spm=a2c6h.13651102.0.0.655e1b11vh8Xuy>

该链接记录了Ubuntu和centos中docker的安装方法，如下所示。

Ubuntu（采用apt-get安装）：

|  |
| --- |
| # step 1: 安装必要的一些系统工具  sudo apt-get update  sudo apt-get -y install apt-transport-https ca-certificates curl software-properties-common  # step 2: 安装GPG证书  curl -fsSL https://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | sudo apt-key add -  # Step 3: 写入软件源信息  sudo add-apt-repository "deb [arch=amd64] https://mirrors.aliyun.com/docker-ce/linux/ubuntu $(lsb\_release -cs) stable"  # Step 4: 更新并安装Docker-CE  sudo apt-get -y update  sudo apt-get -y install docker-ce  # 安装指定版本的Docker-CE:  # Step 1: 查找Docker-CE的版本:  # apt-cache madison docker-ce  # docker-ce | 17.03.1~ce-0~ubuntu-xenial | https://mirrors.aliyun.com/docker-ce/linux/ubuntu xenial/stable amd64 Packages  # docker-ce | 17.03.0~ce-0~ubuntu-xenial | https://mirrors.aliyun.com/docker-ce/linux/ubuntu xenial/stable amd64 Packages  # Step 2: 安装指定版本的Docker-CE: (VERSION例如上面的17.03.1~ce-0~ubuntu-xenial)  # sudo apt-get -y install docker-ce=[VERSION] |

如果是centos7.系列版本（最好centos7.4或以上版本），采用yum命令安装。只下载不安装的命令为:

$ yum install docker-ce --downloaddir=./ --downloadonly

然后可以用yum localinstall 命令来离线安装。

|  |
| --- |
| # step 1: 安装必要的一些系统工具  sudo yum install -y yum-utils device-mapper-persistent-data lvm2  # Step 2: 添加软件源信息  sudo yum-config-manager --add-repo <https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>  # Step 3  sudo sed -i 's+download.docker.com+mirrors.aliyun.com/docker-ce+' /etc/yum.repos.d/docker-ce.repo  # Step 4: 更新并安装Docker-CE  sudo yum makecache fast  sudo yum -y install docker-ce  # Step 4: 开启Docker服务  sudo service docker start  # 注意：  # 官方软件源默认启用了最新的软件，您可以通过编辑软件源的方式获取各个版本的软件包。例如官方并没有将测试版本的软件源置为可用，您可以通过以下方式开启。同理可以开启各种测试版本等。  # vim /etc/yum.repos.d/docker-ce.repo  # 将[docker-ce-test]下方的enabled=0修改为enabled=1  #  # 安装指定版本的Docker-CE:# Step 1: 查找Docker-CE的版本:  # yum list docker-ce.x86\_64 --showduplicates | sort -r  # Loading mirror speeds from cached hostfile  # Loaded plugins: branch, fastestmirror, langpacks# docker-ce.x86\_64 17.03.1.ce-1.el7.centos docker-ce-stable  # docker-ce.x86\_64 17.03.1.ce-1.el7.centos @docker-ce-stable  # docker-ce.x86\_64 17.03.0.ce-1.el7.centos docker-ce-stable  # Available Packages  # Step2: 安装指定版本的Docker-CE: (VERSION例如上面的17.03.0.ce.1-1.el7.centos)  # sudo yum -y install docker-ce-[VERSION] |

这里为了便于后续学习，采用Ubuntu系统。首先安装Ubuntu虚拟机，操作系统为Ubuntu20.04.03 LTS。

安装完操作系统后，用如下截图说明开启root登录ssh方式：



采用apt方式安装docker。docker分为客户端和服务端，分别为docker-ce-cli，docker-ce。

|  |
| --- |
| # sudo apt-get update  # sudo apt-get -y install apt-transport-https ca-certificates curl software-properties-common  # curl -fsSL https://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | sudo apt-key add -  # sudo add-apt-repository "deb [arch=amd64] https://mirrors.aliyun.com/docker-ce/linux/ubuntu $(lsb\_release -cs) stable"  # sudo apt-get -y update  # apt-cache madison docker-ce  # apt install docker-ce=5:20.10.17~3-0~ubuntu-focal docker-ce-cli=5:20.10.17~3-0~ubuntu-focal -y |

安装后结果如下：

|  |
| --- |
| root@ubuntu2004:~# docker info  Client:  Context: default  Debug Mode: false  Plugins:  app: Docker App (Docker Inc., v0.9.1-beta3)  buildx: Docker Buildx (Docker Inc., v0.8.2-docker)  scan: Docker Scan (Docker Inc., v0.17.0)  Server:  Containers: 0  Running: 0  Paused: 0  Stopped: 0  Images: 0  Server Version: 20.10.17  Storage Driver: overlay2  Backing Filesystem: extfs  Supports d\_type: true  Native Overlay Diff: true  userxattr: false  Logging Driver: json-file  Cgroup Driver: cgroupfs  Cgroup Version: 1  Plugins:  Volume: local  Network: bridge host ipvlan macvlan null overlay  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog  Swarm: inactive  Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc  Default Runtime: runc  Init Binary: docker-init  containerd version: 9cd3357b7fd7218e4aec3eae239db1f68a5a6ec6  runc version: v1.1.4-0-g5fd4c4d  init version: de40ad0  Security Options:  apparmor  seccomp  Profile: default  Kernel Version: 5.4.0-131-generic  Operating System: Ubuntu 20.04.3 LTS  OSType: linux  Architecture: x86\_64  CPUs: 2  Total Memory: 1.907GiB  Name: ubuntu2004  ID: NQYW:HVQL:6SMC:ECSE:LCNN:P3TK:CIVK:3FKJ:26WQ:QOPU:RRK7:V2CM  Docker Root Dir: /var/lib/docker  Debug Mode: false  Registry: https://index.docker.io/v1/  Labels:  Experimental: false  Insecure Registries:  127.0.0.0/8  Live Restore Enabled: false  WARNING: No swap limit support  root@ubuntu2004:~# |

# 使用docker卷

运行docker容器时，如果想让数据做持久化，通常可以采用数据卷的方式，将目录映射到容器中。当然也可以将文件映射到容器中，例如不采用镜像里的某个配置文件，此时可以将配置文件编辑好，放到宿主机上某个路径，然后以数据卷挂载方式将文件映射到容器里指定路径即可。

配置镜像加速：

|  |
| --- |
| # 新建daemon.json文件，然后重启docker服务  root@ubuntu2004:~# cat /etc/docker/daemon.json  {  "registry-mirrors": ["https://9916w1ow.mirror.aliyuncs.com"],  "log-opts": {  "max-file": "5",  "max-size": "100m"  }  }  root@ubuntu2004:~# systemctl stop docker.socket  root@ubuntu2004:~# systemctl start docker.socket |

下载nginx和mysql镜像：

|  |
| --- |
| root@ubuntu2004:~# docker pull nginx:1.20.2  1.20.2: Pulling from library/nginx  a2abf6c4d29d: Pull complete  da03644a1293: Pull complete  dcbfc6badd70: Pull complete  3f7ccff97047: Pull complete  49e31097680b: Pull complete  c423e1dacb26: Pull complete  Digest: sha256:03f3cb0afb7bd5c76e01bfec0ce08803c495348dccce37bcb82c347b4853c00b  Status: Downloaded newer image for nginx:1.20.2  docker.io/library/nginx:1.20.2  root@ubuntu2004:~# docker pull nginx:1.20.2  1.20.2: Pulling from library/nginx  a2abf6c4d29d: Pull complete  da03644a1293: Pull complete  dcbfc6badd70: Pull complete  3f7ccff97047: Pull complete  49e31097680b: Pull complete  c423e1dacb26: Pull complete  Digest: sha256:03f3cb0afb7bd5c76e01bfec0ce08803c495348dccce37bcb82c347b4853c00b  Status: Downloaded newer image for nginx:1.20.2  docker.io/library/nginx:1.20.2 |

生产中的运行容器，我们可以编辑好yaml文件，然后采用docker-compose指定yaml文件启动容器。如果是临时容器，可以用docker run命令启动运行，加上--rm参数，运行完程序即退出。

nginx一般用作反向代理和作为web服务器，前端代码目录可以以数据卷的方式挂载到nginx中，这里不做演示。我这里只下载镜像，后面有需要再使用此镜像。

mysql容器运行，数据不能随容器的删除而删除，所以需要将数据目录落盘。下面将数据存放到/data/mysql目录下。

|  |
| --- |
| # 1. 创建mysql容器，密码为12345678，数据落盘到/data/mysql,端口映射为3306  root@ubuntu2004:~# docker run -it -p 3306:3306 -d -v /data/mysql:/var/lib/mysql -e MYSQL\_ROOT\_PASSWORD=12345678 --name mysql mysql:5.6.48  3502c58b797b2a09299f9fe2950472bd57fba70a1a6b732f636f4abb16eadf4c  # 2. 检查mysql容器为up状态。  root@ubuntu2004:~# docker ps -a  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  3502c58b797b mysql:5.6.48 "docker-entrypoint.s…" 5 seconds ago Up 4 seconds 0.0.0.0:3306->3306/tcp, :::3306->3306/tcp mysql  #3. 进入mysql容器，访问mysql正常。  root@ubuntu2004:~# docker exec -it mysql bash  root@3502c58b797b:/# mysql -uroot -p12345678 -h127.0.0.1  Warning: Using a password on the command line interface can be insecure.  Welcome to the MySQL monitor. Commands end with ; or \g.  Your MySQL connection id is 1  Server version: 5.6.48 MySQL Community Server (GPL)  Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.  Oracle is a registered trademark of Oracle Corporation and/or its  affiliates. Other names may be trademarks of their respective  owners.  Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  mysql> show databases;  +--------------------+  | Database |  +--------------------+  | information\_schema |  | mysql |  | performance\_schema |  +--------------------+  3 rows in set (0.00 sec)  mysql> |

# 使用Docker的bridge和container模式网络

docker run方式运行容器，如果不指定--net选项，默认就是采用bridge网络，使用的docker0网桥。

|  |
| --- |
| # 新建nginx容器，采用bridge模式网络。  root@ubuntu2004:~# docker run -d -p 80:80 --name nginx nginx:1.23.1-alpine  12cd50e37386aaec9e0917873a55830de7128ab9ac11d2190ca54333a1ae8696  root@ubuntu2004:~# docker ps -a  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  12cd50e37386 nginx:1.23.1-alpine "/docker-entrypoint.…" 4 seconds ago Up 3 seconds 0.0.0.0:80->80/tcp, :::80->80/tcp nginx  3502c58b797b mysql:5.6.48 "docker-entrypoint.s…" 10 minutes ago Up 10 minutes 0.0.0.0:3306->3306/tcp, :::3306->3306/tcp mysql  # 可以看出nginx容器的ip为 172.17.0.3，与docker0同一网段。  root@ubuntu2004:~# docker exec nginx ip a  1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  inet 127.0.0.1/8 scope host lo  valid\_lft forever preferred\_lft forever  10: eth0@if11: <BROADCAST,MULTICAST,UP,LOWER\_UP,M-DOWN> mtu 1500 qdisc noqueue state UP  link/ether 02:42:ac:11:00:03 brd ff:ff:ff:ff:ff:ff  inet 172.17.0.3/16 brd 172.17.255.255 scope global eth0  valid\_lft forever preferred\_lft forever  # 宿主机的ip信息，docker0网桥的ip为172.17.0.1  root@ubuntu2004:~# ip a  1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  inet 127.0.0.1/8 scope host lo  valid\_lft forever preferred\_lft forever  inet6 ::1/128 scope host  valid\_lft forever preferred\_lft forever  2: ens33: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc fq\_codel state UP group default qlen 1000  link/ether 00:0c:29:83:39:a1 brd ff:ff:ff:ff:ff:ff  inet 10.0.0.129/24 brd 10.0.0.255 scope global dynamic ens33  valid\_lft 1471sec preferred\_lft 1471sec  inet6 fe80::20c:29ff:fe83:39a1/64 scope link  valid\_lft forever preferred\_lft forever  3: docker0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UP group default  link/ether 02:42:9c:b1:52:64 brd ff:ff:ff:ff:ff:ff  inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0  valid\_lft forever preferred\_lft forever  inet6 fe80::42:9cff:feb1:5264/64 scope link  valid\_lft forever preferred\_lft forever |

container模式网络，是与指定的容器同一网络ip。

|  |
| --- |
| #1. 指定php容器的网络为container模式网络，和nginx在同一网络命名空间。  root@ubuntu2004:~# docker run -it -d --name php-container --net=container:nginx --name=php php:7.4.30-fpm-alpine  Unable to find image 'php:7.4.30-fpm-alpine' locally  7.4.30-fpm-alpine: Pulling from library/php  213ec9aee27d: Already exists  a600fdbc30cc: Pull complete  0cdd6cb15c0d: Pull complete  8a4c40d8aee7: Pull complete  77e67522f4fd: Pull complete  d181492ef8e9: Pull complete  b3ee11034df4: Pull complete  eb72a5cacfff: Pull complete  9acd67bcb441: Pull complete  d2d9674ad948: Pull complete  Digest: sha256:7e551f8b4666967229d03a1f00008ed5098d5cccbf8a59129203539ff745ecbb  Status: Downloaded newer image for php:7.4.30-fpm-alpine  459b33f828cedf14a4ed04a7104a3b7c008f55dc618845d6d3f66760d5265048  # nginx的ip为172.17.0.3  root@ubuntu2004:~# docker exec -it nginx ip a  1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  inet 127.0.0.1/8 scope host lo  valid\_lft forever preferred\_lft forever  10: eth0@if11: <BROADCAST,MULTICAST,UP,LOWER\_UP,M-DOWN> mtu 1500 qdisc noqueue state UP  link/ether 02:42:ac:11:00:03 brd ff:ff:ff:ff:ff:ff  inet 172.17.0.3/16 brd 172.17.255.255 scope global eth0  valid\_lft forever preferred\_lft forever  # php的ip为172.17.0.3  root@ubuntu2004:~# docker exec -it php ip a  1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  inet 127.0.0.1/8 scope host lo  valid\_lft forever preferred\_lft forever  10: eth0@if11: <BROADCAST,MULTICAST,UP,LOWER\_UP,M-DOWN> mtu 1500 qdisc noqueue state UP  link/ether 02:42:ac:11:00:03 brd ff:ff:ff:ff:ff:ff  inet 172.17.0.3/16 brd 172.17.255.255 scope global eth0  valid\_lft forever preferred\_lft forever |