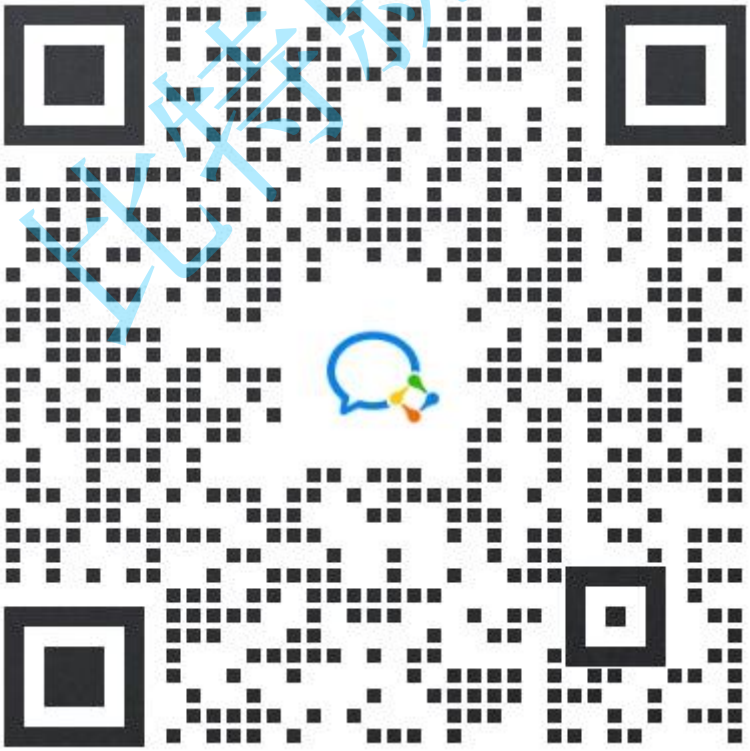
LXC 容器实战

版权说明

对比特项目感兴趣，可以联系这个微信。



LXC 容器操作实战

实战目的

通过 lxc 来完成容器的创建，体会容器，并了解 docker 并不是容器的唯一实现。自 docker 0.9 版本起，docker 除了继续支持 LXC 外，还开始引入自家的 libcontainer，试图打造更通用的底层容器虚拟化库。如今的 docker 基本上都已经是使用 libcontainer 而非 LXC 了。

基础知识

LXC 的常用命令如下：

lxc-checkconfig

检查系统环境是否满足容器使用要求；

格式：lxc-checkconfig

lxc-create

创建 lxc 容器；

格式：lxc-create -n NAME -t TEMPLATE_NAME [-- template-options]

lxc-start

启动容器；

格式：lxc-start -n NAME -d

lxc-ls

列出所有容器,-f 表示打印常用的信息；

格式：lxc-ls -f

lxc-info

查看容器相关的信息；

格式：lxc-info -n NAME

lxc-attach

进入容器执行命令；

格式：lxc-attach --name=NAME [-- COMMAND]

lxc-stop

停止容器;

格式：lxc-stop -n NAME

lxc-destory

删除处于停机状态的容器;

格式：lxc-destory -n NAME

安装 LXC

Ubuntu 安装

安装前执行检查看下是否需要卸载，如果需要卸载，执行下面的命令完成卸载，不需要直接到第 2 步

```Shell
# 一、检查是否安装。清理资源
systemctl status lxc
lxc-stop -n xxx # lxc-ls -f 遍历所有容器,停止运行的容器
lxc-destroy -n xxx # 删除对应的容器
# 二、 卸载软件
apt-get purge --auto-remove lxc lxc-templates
# 三、 检查服务已经没有该服务了
systemctl status lxc
```

没有安装的话，执行下面的命令完成安装

```Shell
#一、安装
#lxc            主程序包
#lxc-templates lxc 的配置模板
#bridge-utils  网桥管理工具
apt install lxc lxc-templates bridge-utils -y

#二、检查服务是否正常运行
systemctl status lxc
```

CentOS 安装

安装前执行检查看下是否需要卸载，如果需要卸载，执行下面的命令完成卸载，不需要直接到第 2 步

```
Shell
# 一、检查是否安装。清理资源
systemctl status lxc #检查是否安装
lxc-stop -n xxx # lxc-ls -f 遍历所有容器,停止对应的容器
lxc-destroy -n xxx #删除对应的容器
# 二、 卸载软件
yum remove lxc  lxc-templates  lxc-libs lxc-extra libvirt
debootstrap
# 三、检查，提示服务不存在
systemctl status lxc
```

CentOS 安装 LXC，如果已经安装，可以检查下是否需要卸载，如果需要卸载执行 Centos 卸载 LXC

```
Shell
# 一、 配置源
yum -y install epel-release          #这个软件包里包含 epel yum 源和
GPG 的配置

# 二、 安装程序
# lxc 主程序包
# lxc-templates lxc 的配置模板
# bridge-utils 网桥管理工具 lxc-libs lxc 所需的库文件
# libcgroup cgroup 安装包
# libvirt 管理 Linux 的虚拟化功能所需的服务器端守护程序。 需要针对特定驱
动程序的管理程序。
# debootstrap debootstrap 是 Debian 引导程序,它允许您将 Debian 基本系统
(例如 Debian 或 Ubuntu)安装到当前正在运行的系统的目录中。
yum -y install lxc lxc-templates bridge-utils lxc-libs libcgroup
libvirt lxc-extra debootstrap

#三、启动和检查
#如果未运行输入以下命令完成启动
systemctl start lxc #启动 lxc 服务
systemctl start libvirtd #启动虚拟机监控服务
systemctl status lxc
systemctl status libvirtd
```

LXC 容器操作实战

1. 检查 lxc 是否运行

```
Shell
root@139-159-150-152:/var/run/docker/netns# systemctl status lxc
● lxc.service - LXC Container Initialization and Autoboot Code
     Loaded: loaded (/lib/systemd/system/lxc.service; enabled;
vendor preset: enabled)
     Active: active (exited) since Fri 2023-03-17 11:27:47 CST;
1min 33s ago
       Docs: man:lxc-autostart
             man:lxc
   Main PID: 254137 (code=exited, status=0/SUCCESS)
      Tasks: 0 (limit: 2274)
     Memory: 0B
     CGroup: /system.slice/lxc.service

Mar 17 11:27:47 139-159-150-152 systemd[1]: Starting LXC Container
Initialization and Autoboot Code...
Mar 17 11:27:47 139-159-150-152 systemd[1]: Finished LXC Container
Initialization and Autoboot Code.
```

2. 检查 lxc 的功能支持情况

```
Shell
root@139-159-150-152:/var/run/docker/netns#  lxc-checkconfig
LXC version 4.0.12
Kernel configuration not found at /proc/config.gz; searching...
Kernel configuration found at /boot/config-5.4.0-100-generic
--- Namespaces ---
Namespaces: enabled
Utsname namespace: enabled
Ipc namespace: enabled
Pid namespace: enabled
User namespace: enabled
Network namespace: enabled

--- Control groups ---
Cgroups: enabled
Cgroup namespace: enabled

Cgroup v1 mount points:
```

```
/sys/fs/cgroup/systemd
/sys/fs/cgroup/cpu,cpuacct
/sys/fs/cgroup/perf_event
/sys/fs/cgroup/net_cls,net_prio
/sys/fs/cgroup/pids
/sys/fs/cgroup/blkio
/sys/fs/cgroup/freezer
/sys/fs/cgroup/cpuset
/sys/fs/cgroup/hugetlb
/sys/fs/cgroup/memory
/sys/fs/cgroup/rdma
/sys/fs/cgroup/devices

Cgroup v2 mount points:
/sys/fs/cgroup/unified

Cgroup v1 clone_children flag: enabled
Cgroup device: enabled
Cgroup sched: enabled
Cgroup cpu account: enabled
Cgroup memory controller: enabled
Cgroup cpuset: enabled

--- Misc ---
Veth pair device: enabled, loaded
Macvlan: enabled, not loaded
Vlan: enabled, not loaded
Bridges: enabled, loaded
Advanced netfilter: enabled, not loaded
CONFIG_IP_NF_TARGET_MASQUERADE: enabled, not loaded
CONFIG_IP6_NF_TARGET_MASQUERADE: enabled, not loaded
CONFIG_NETFILTER_XT_TARGET_CHECKSUM: enabled, loaded
CONFIG_NETFILTER_XT_MATCH_COMMENT: enabled, not loaded
FUSE (for use with lxcfs): enabled, not loaded

--- Checkpoint/Restore ---
checkpoint restore: enabled
CONFIG_FHANDLE: enabled
CONFIG_EVENTFD: enabled
CONFIG_EPOLL: enabled
CONFIG_UNIX_DIAG: enabled
CONFIG_INET_DIAG: enabled
CONFIG_PACKET_DIAG: enabled
CONFIG_NETLINK_DIAG: enabled
```

```
File capabilities:

Note : Before booting a new kernel, you can check its
configuration
usage : CONFIG=/path/to/config /usr/bin/lxc-checkconfig
```

3. 查看 lxc 提供的容器模板

```
Shell
root@139-159-150-152:/var/run/docker/netns#  ls
/usr/share/lxc/templates/
lxc-alpine     lxc-busybox  lxc-debian    lxc-fedora-legacy  lxc-
oci           lxc-oracle  lxc-sabayon     lxc-sshd          lxc-
voidlinux
lxc-altlinux   lxc-centos   lxc-download  lxc-gentoo         lxc-
openmandriva  lxc-plamo   lxc-slackware   lxc-ubuntu
lxc-archlinux  lxc-cirros   lxc-fedora    lxc-local          lxc-
opensuse      lxc-pld     lxc-sparclinux  lxc-ubuntu-cloud
```

4. 创建一个 lxc 虚拟主机，这个命令就会下载安装指定环境下的软件包，创建新容器。整个过程需要时间较长，与容器的类型有关。

```
Shell
#创建 Ubuntu LXC 容器，-t 指定模板容器，-n 指定要创建的容器名，下面创建
的是 ubuntu
#Centos 上创建 centos 的命令：lxc-create -t centos --name centos1 --
--release 7 --arch x86_64
#Ubuntu 上创建 centos 的命令，注意模板需要使用 download：lxc-create --
name centos7 --template=download -- --dist=centos --release=7 --
arch=amd64

root@139-159-150-152:/var/run/docker/netns# lxc-create -t ubuntu -
-name lxchost1 -- -r xenial -a amd64
#

#创建完成显示

##
# The default user is 'ubuntu' with password 'ubuntu'!
# Use the 'sudo' command to run tasks as root in the container.
##
```

5. 下载安装完所有软件包后，LXC 容器镜像就创建完成了，你可以看到默认的登录界面。容器被放到 /var/lib/lxc/<容器名> 这个目录下，容器的根文件系统放在 /var/lib/lxc/<容器名>/rootfs 目录下。创建过程中下载的软件包保存在 /var/cache/lxc 目录下面，当你想另外建一个一样的容器时，可以省去很多下载时间。

```
Shell
root@139-159-150-152:/var/run/docker/netns# ll
/var/lib/lxc/lxchost1/
total 16
drwxrwx---  3 root root 4096 Mar 17 11:44 ./
drwx------  3 root root 4096 Mar 17 11:34 ../
-rw-r-----  1 root root  679 Mar 17 11:44 config
drwxr-xr-x 17 root root 4096 Mar 17 11:42 rootfs/
root@139-159-150-152:/var/run/docker/netns# ll /var/cache/lxc/
total 12
drwx------  3 root root 4096 Mar 17 11:34 ./
drwxr-xr-x 19 root root 4096 Mar 17 11:27 ../
drwxr-xr-x  3 root root 4096 Mar 17 11:44 xenial/
```

6. 查看创建的容器信息。

```
Shell
root@139-159-150-152:/var/run/docker/netns# lxc-ls -f
NAME     STATE   AUTOSTART GROUPS IPV4 IPV6 UNPRIVILEGED
lxchost1 STOPPED 0         -      -    -    false
```

7. 启动容器，我们可以看到容器状态为运行中

```
Shell
root@139-159-150-152:/var/run/docker/netns# lxc-start -n lxchost1
-d
root@139-159-150-152:/var/run/docker/netns# lxc-ls -f
NAME     STATE   AUTOSTART GROUPS IPV4       IPV6 UNPRIVILEGED
lxchost1 RUNNING 0         -      10.0.3.248 -    false
```

8. 查看容器的详细信息

```
Shell
root@139-159-150-152:/var/run/docker/netns# lxc-info -n lxchost1
Name:           lxchost1
```

```
State:          RUNNING
PID:            282127
IP:             10.0.3.248
CPU use:        0.59 seconds
BlkIO use:      29.45 MiB
Memory use:     59.52 MiB
KMem use:       6.82 MiB
Link:           vethbg8LKH
 TX bytes:      1.73 KiB
 RX bytes:      6.61 KiB
 Total bytes:   8.33 KiB
```

9. 容器 ip 为 10.0.3.248，我们通过 ssh 进入容器，查看 ip 地址，磁盘挂载信息，目录信息和宿主机都不一样

```
Shell
root@139-159-150-152:/var/run/docker/netns# ssh ubuntu@10.0.3.248
ubuntu@10.0.3.248's password:


ubuntu@lxchost1:~$ ip addr
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: eth0@if562: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
noqueue state UP group default qlen 1000
    link/ether 00:16:3e:71:d8:3d brd ff:ff:ff:ff:ff:ff link-
netnsid 0
    inet 10.0.3.248/24 brd 10.0.3.255 scope global dynamic eth0
       valid_lft 2844sec preferred_lft 2844sec
    inet6 fe80::216:3eff:fe71:d83d/64 scope link
       valid_lft forever preferred_lft forever
ubuntu@lxchost1:~$ uname -a
Linux lxchost1 5.4.0-100-generic #113-Ubuntu SMP Thu Feb 3
18:43:29 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
ubuntu@lxchost1:~$ ll /
total 52
drwxr-xr-x  17 root root 4096 Mar 17 03:51 ./
drwxr-xr-x  17 root root 4096 Mar 17 03:51 ../
```

```
lrwxrwxrwx   1 root root    7 Mar 17 03:40 bin -> usr/bin/
drwxr-xr-x   2 root root 4096 Apr 15  2020 boot/
drwxr-xr-x   7 root root  540 Mar 17 03:51 dev/
drwxr-xr-x  63 root root 4096 Mar 17 03:51 etc/
drwxr-xr-x   3 root root 4096 Mar 17 03:44 home/
lrwxrwxrwx   1 root root    7 Mar 17 03:40 lib -> usr/lib/
lrwxrwxrwx   1 root root    9 Mar 17 03:40 lib32 -> usr/lib32/
lrwxrwxrwx   1 root root    9 Mar 17 03:40 lib64 -> usr/lib64/
lrwxrwxrwx   1 root root   10 Mar 17 03:40 libx32 -> usr/libx32/
drwxr-xr-x   2 root root 4096 Mar 17 03:40 media/
drwxr-xr-x   2 root root 4096 Mar 17 03:40 mnt/
drwxr-xr-x   2 root root 4096 Mar 17 03:40 opt/
dr-xr-xr-x 225 root root    0 Mar 17 03:51 proc/
drwx------   2 root root 4096 Mar 17 03:40 root/
drwxr-xr-x  13 root root  420 Mar 17 04:04 run/
lrwxrwxrwx   1 root root    8 Mar 17 03:40 sbin -> usr/sbin/
drwxr-xr-x   2 root root 4096 Mar 17 03:40 srv/
dr-xr-xr-x  13 root root    0 Mar 17 03:51 sys/
drwxrwxrwt   9 root root 4096 Mar 17 04:04 tmp/
drwxr-xr-x  13 root root 4096 Mar 17 03:40 usr/
drwxr-xr-x  11 root root 4096 Mar 17 03:40 var/
ubuntu@lxchost1:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1        40G   21G   17G  57% /
none            492K  4.0K  488K   1% /dev
tmpfs           992M     0  992M   0% /dev/shm
tmpfs           199M  108K  199M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           992M     0  992M   0% /sys/fs/cgroup
tmpfs           199M     0  199M   0% /run/user/1000
ubuntu@lxchost1:~$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 03:51 ?        00:00:00 /sbin/init
root          44       1  0 03:51 ?        00:00:00
/lib/systemd/systemd-journald
systemd+      72       1  0 03:51 ?        00:00:00
/lib/systemd/systemd-networkd
root          76       1  0 03:51 ?        00:00:00 /usr/sbin/cron
-f
message+      77       1  0 03:51 ?        00:00:00 /usr/bin/dbus-
daemon --system --address=systemd: --nofork --nopidfile --systemd-
activation --syslog-only
root          79       1  0 03:51 ?        00:00:00
/usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-
```

```
triggers
syslog        80       1  0 03:51 ?        00:00:00
/usr/sbin/rsyslogd -n -iNONE
root          81       1  0 03:51 ?        00:00:00
/lib/systemd/systemd-logind
systemd+      82       1  0 03:51 ?        00:00:00
/lib/systemd/systemd-resolved
root          88       1  0 03:51 pts/0    00:00:00 /sbin/agetty -
o -p -- \u --noclear --keep-baud console 115200,38400,9600 vt220
root          89       1  0 03:51 ?        00:00:00 sshd:
/usr/sbin/sshd -D [listener] 0 of 10-100 startups
root         163      89  0 04:03 ?        00:00:00 sshd: ubuntu
[priv]
ubuntu       166       1  0 04:04 ?        00:00:00
/lib/systemd/systemd --user
ubuntu       167     166  0 04:04 ?        00:00:00 (sd-pam)
ubuntu       182     163  0 04:04 ?        00:00:00 sshd:
ubuntu@pts/5
ubuntu       183     182  0 04:04 pts/5    00:00:00 -bash
ubuntu       196     183  0 04:04 pts/5    00:00:00 ps -ef
```

10. 在容器外面执行命令

```
Shell
root@139-159-150-152:/var/run/docker/netns# lxc-attach -n lxchost1
--clear-env -- echo "Hello bit"
Hello bit
```

11. 停止容器

```
Shell
root@139-159-150-152:/var/run/docker/netns# lxc-stop -n lxchost1
root@139-159-150-152:/var/run/docker/netns# lxc-ls -f
NAME     STATE   AUTOSTART GROUPS IPV4 IPV6 UNPRIVILEGED
lxchost1 STOPPED 0         -      -    -    false
```

12. 删除容器

```
Shell
root@139-159-150-152:/var/run/docker/netns# lxc-destroy -n
lxchost1
root@139-159-150-152:/var/run/docker/netns# lxc-ls -f
root@139-159-150-152:/var/run/docker/netns#
```