

容器和虚拟网络



齐划一



qi@huayi.email



山东大学

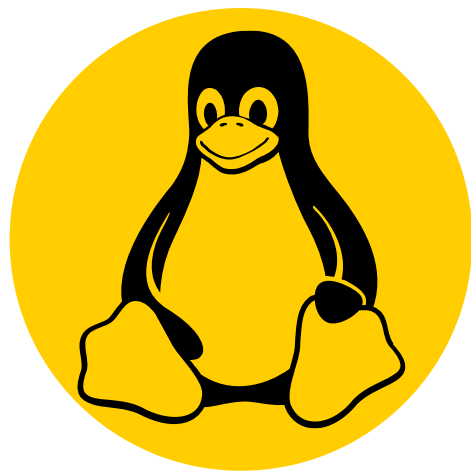


内容提要

- 容器的应用场景
- 常见的容器技术
- IPv4 网络
- IPv6 网络

1

容器的应用场景



Linux 能干什么？



Linux 能干什么？

- 作为生产环境，运行各种 Server；
- 作为开发环境，尽可能贴近生产环境，进行编程；
- 科研计算；
- 通过编写脚本，完成各种自动工作；
-



生产环境

包管理器

依赖项多，版本易冲突，软件过新或过旧，多个程序可能要求不同的库版本……

编译安装

编译困难，安装后极难清除，软件几乎不能升级，编译环境依然依赖包管理器……

```
user@ubuntu: ~  
user@ubuntu:~$ sudo apt install php  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  apache2 apache2-bin apache2-data apache2-utils libapache2-mod-php7.4  
  libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.2-0  
  php-common php7.4 php7.4-cli php7.4-common php7.4-json php7.4-opcache  
  php7.4-readline  
Suggested packages:  
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom php-pear  
The following NEW packages will be installed:  
  apache2 apache2-bin apache2-data apache2-utils libapache2-mod-php7.4  
  libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.2-0  
  php php-common php7.4 php7.4-cli php7.4-common php7.4-json php7.4-opcache  
  php7.4-readline  
0 upgraded, 18 newly installed, 0 to remove and 1 not upgraded.  
Need to get 5,838 kB of archives.  
After this operation, 25.9 MB of additional disk space will be used.  
Do you want to continue? [Y/n]
```



开发环境

- 编程工具链版本号要求，尽可能与生产环境保持一致；
- 同时开发多个项目时的版本号冲突；
- C/C++ 库管理通过系统的包管理器完成，难以控制版本；
- PHP 等语言依赖项过多过杂；
- MongoDB 等软件对开源极不友好，不包含在系统源中；
-

```
CMakeLists.txt - libcvd - VSCode
File Edit Selection View Go Run Terminal Help

EXPLORER
LIBCVD
  .github
  ad_hoc_tests
  cmake
  cvd
  cvd_src
  doc
  examples
  m4
  progs
  tests
  .clang-format
  .gitignore
  a.out
  Authors
  CMakeLists.txt
  config.guess
  config.log
  config.status
  config.sub
  configure
  configure_osx_32bit
  configure.ac
  OUTLINE

CMakeLists.txt
1 cmake_minimum_required(VERSION 3.10)
2 project(CVD)
3 enable_testing()
4
5 option(CVD_ENABLE_TESTS "Build libCVD tests" ON)
6 option(CVD_ENABLE_PROGS "Build libCVD programs" ON)
7 option(CVD_ENABLE_EXAMPLES "Build libCVD examples" ON)
8
9 include(TestBigEndian)
10 include(CheckSymbolExists)
11
12 set(CMAKE_DEBUG_POSTFIX _debug)
13
14 # Dependencies that can be automatically found by CMake.
15 include(cmake/CVDFindAllDeps.cmake)
16
17 # Detect endianness
18
19 TEST_BIG_ENDIAN(CVD_INTERNAL_ARCH_BIG_ENDIAN)
20 if(CVD_INTERNAL_ARCH_BIG_ENDIAN)
21   set(CVD_INTERNAL_ARCH_LITTLE_ENDIAN OFF)
22 else()
23   set(CVD_INTERNAL_ARCH_LITTLE_ENDIAN ON)
24 endif()
25
26 # Basic source files and headers for all platforms and options.
27
```



科研计算

- 需要在多个服务器部署相同的程序；
- 往往和其他人分时共享服务器；按需使用，时而占用大量资源，时而空闲；
- 每个程序的环境要求不同；
- 每隔一段时间，论文实验部分又要返工重做；
-

```
bob@bob-os-1:~$ nvidia-smi
Fri Jun  4 15:13:22 2021

+-----+
| NVIDIA-SMI 440.44           Driver Version: 440.44           CUDA Version: N/A           |
+-----+-----+-----+-----+-----+-----+
| GPU   Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+-----+-----+-----+-----+
|    0   GeForce GTX 108...    Off      | 00000000:02:00.0 Off |           N/A       |
| 11%    27C    P0      55W / 250W |  0MiB / 11178MiB |      0%      Default |
+-----+-----+-----+-----+-----+-----+
|    1   GeForce GTX 108...    Off      | 00000000:82:00.0 Off |           N/A       |
| 12%    27C    P0      51W / 250W |  0MiB / 11178MiB |      0%      Default |
+-----+-----+-----+-----+-----+-----+

+-----+
| Processes:                                                       GPU Memory |
|  GPU       PID    Type    Process name                     Usage    |
+-----+-----+-----+-----+-----+-----+
| No running processes found                                     |
+-----+

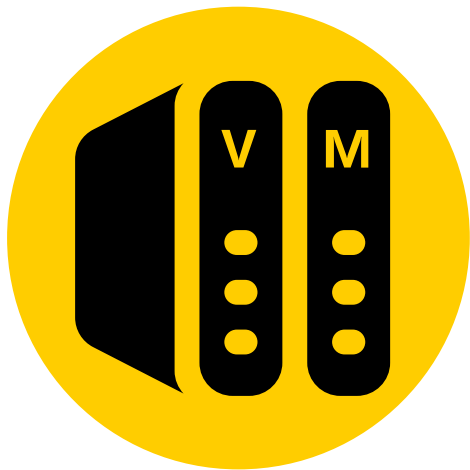
[bob@bob-os-1 ~]$
```




脚本

- 往往会对各种程序的命令行输出进行解析处理，对版本号要求极高，便携性差；
- 脚本依赖各种程序，系统很可能没有安装这些程序；
- DevOps 下的自动构建、持续交付等功能需要程序员提供稳定的编译环境；
- 脚本编写出错时可能会导致极度危险的行为；
-

```
user@DESKTOP-9400F ~$ ./sduelec
sduelec v0.1
Electricity Bill Query for SDU-Qingdao
输入学工号: 20
学工号: 20201
姓名:
卡号:
[0] 威海缴电费
[1] 锐泰电控
[2] 青岛电控
请选择: 2
[0] 青岛校区
请选择: 0
[0] 凤凰居6号楼
[1] T1
[2] S1一多书院
[3] S11
[4] 凤凰居9号楼
[5] S5凤凰居5号楼
[6] S2从文书院
[7] T3
[8] S9凤凰居9号楼
[9] S7凤凰居7号楼
[10] S10凤凰居10号楼
[11] S8凤凰居8号楼
[12] 专家公寓2号楼
[13] 凤凰居4号楼
[14] S6凤凰居6号楼
请选择: 0
输入房间号 (例: A101):
房间当前剩余电量90.88
user@DESKTOP-9400F ~$ ./sduelec
房间当前剩余电量90.88
user@DESKTOP-9400F ~$ |
```



虚拟机

隔离性好
通用性强
易于管理



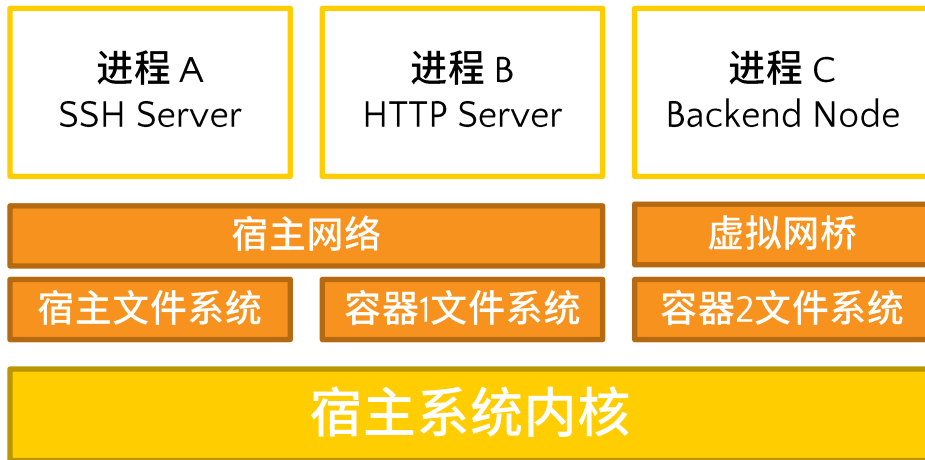
容器

有点像虚拟机
无性能损失
部署迅速



容器

示例



2

常见的容器技术



常见的容器技术

● Docker / Podman

应用最广泛的容器产品，每个容器仅运行一种应用，容器镜像便于制作、传输，易部署。运维神器。Podman 是更好的 Docker 替代品，为 Kubernetes 而生。

● LXC

最像传统的虚拟机，每个容器运行一个较为完整的 Linux 发行版。

● systemd-nspawn

和 LXC 类似，但也像一个大号的 chroot。

● Jail

谁说只有 Linux 才有容器？这是 FreeBSD 下的容器，像沙箱一样，侧重于安全。

```
user@DESKTOP-9400F: ~  
user@DESKTOP-9400F:~$ cat /etc/containers/registries.conf | grep -v '^#'  
unqualified-search-registries = ["docker.io"]  
user@DESKTOP-9400F:~$ sudo podman 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/  
  
user@DESKTOP-9400F:~$ |
```

3

IPv4 网络



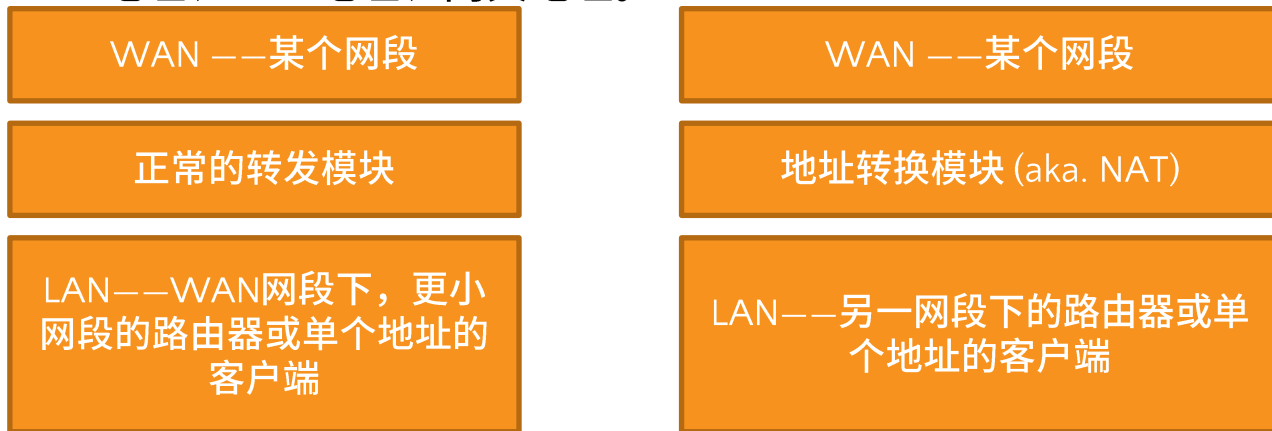
IPv4 网络

- 每个容器要么直接使用宿主机的网络，要么需要连接到虚拟网络。Docker、KVM 等方案提供了零配置的 IPv4 内网方案，开箱即用。
- 需要自定义虚拟网络的场景：
 - 需要 IPv6
 - 自动配置的 IPv4 网段不合适
 - 需要更复杂的 IP 分配、端口转发规则、UPnP 等特性
 - 通过隧道技术为多个物理机的容器提供统一的内网



IPv4 网络

- 家用路由器通过 NAT 技术为下级设备提供网络，通过 DHCPv4 协议分配 IPv4 地址、DNS地址、网关地址。



IPv4 运营商的路由器
IPv6 运营商的路由器
IPv6 用户的路由器

IPv4 用户的路由器
极个别 IPv6 用户的路由器



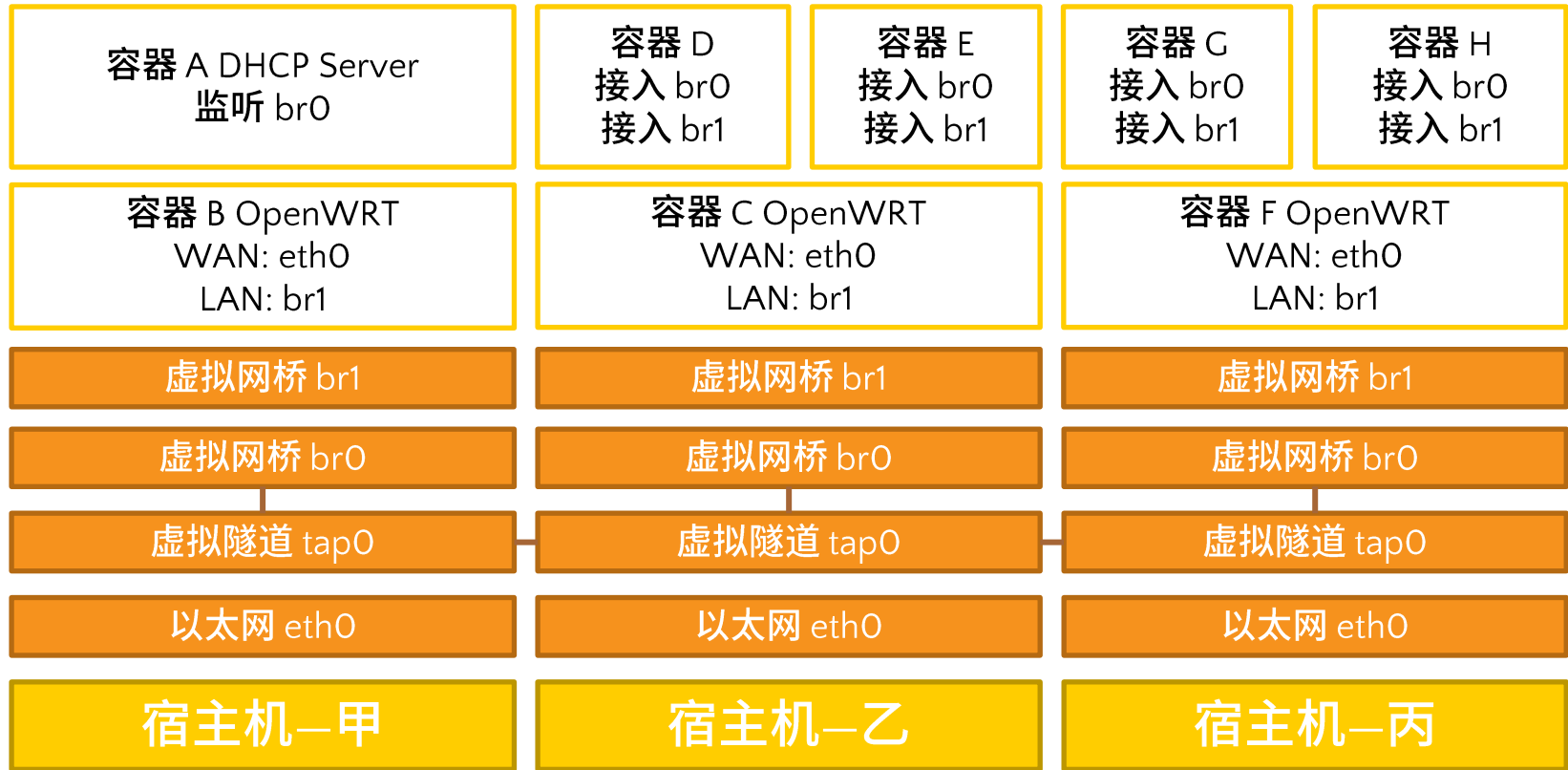
例：打通多物理机容器内网





例：打通多物理机容器内网





4

IPv6 网络



IPv6 网络

- IPv6 的特点：让几乎所有设备地址达到全球可访问的目的。极其适合 P2P 应用。
- 路由器通过 DHCPv6-PD 协议向上获得 IPv6 网段；
- 通过 RA 协议通告下级网关地址等信息；
- 通过 SLAAC 算法 和/或 DHCPv6 协议让下级设备获得 IPv6 地址；
- 通过 RA 协议 和/或 DHCPv6 协议让下级设备获得 DNS 地址。

IPv6 自动配置

☉ ICMPv6: **Router Advertisement** (获得网关地址的唯一方式) , aka. **RA**
RA 包含 IPv6 前缀、网关地址 (可选)、DNS 地址 (可选) 等, 必须设置过期时间

☉ 不考虑 DHCPv6-PD, IPv6 地址共有四套分配方式:

☉ (分配方式一) **SLAAC**

通过 IPv6 前缀和本机 MAC 地址计算出 IPv6 地址, aka. **SLAAC**

DNS 地址等需要通过解析 RA 的 RDNSS 选项获得
(Windows 10 1703 之前的版本 (不含) 不支持从 RA 中获得 DNS 地址)

☉ (分配方式二) **DHCPv6 (stateful)**

DHCPv6 负责分配 IPv6, 并告知 DNS 地址、NTP 地址等 (不包含网关地址)

Android 坚决不支持 DHCPv6, 下同

☉ (分配方式三) **DHCPv6 (stateless) + SLAAC**

通过 SLAAC 计算出 IPv6 地址, DHCPv6 仅负责告知 DNS 地址、NTP 地址等

☉ (分配方式四) **DHCPv6 (stateful) + SLAAC**

SLAAC 和 DHCPv6 (stateful) 的结合, 客户端有两套 IPv6 地址, 爱用哪个用哪个



DHCPv6-PD

普通的 IPv6 客户端：

➤ 1. RA 广播表示支持 SLAAC

——客户端：我自己算出一个 IPv6 地址

➤ 2. RA 广播表示支持 DHCPv6 (stateful) ，前缀长度是 /64

——客户端：你好，DHCPv6 服务器，请给我一个 IPv6 地址

——DHCPv6 服务器：OK，地址是 2001:dead:beef::100/128



DHCPv6-PD

◎ 普通的 IPv6 路由器：

➤ RA 广播表示支持 DHCPv6 (stateful)，前缀长度是 /64

——子 IPv6 路由器：你好，DHCPv6 服务器，请给我一个 /80 的子网

——DHCPv6 服务器：OK，前缀是2001:dead:beef:babe:100::/80

——IPv6 路由器：设置路由表，若目标地址的前缀为该前缀，则转发给该子路由器



然而，

◎ 另一种情况：

➤ RA 广播表示支持 DHCPv6 (stateful)，前缀长度是 /64

——子 IPv6 路由器：你好，DHCPv6 服务器，请给我一个 /80 的子网

——DHCPv6 服务器：滚

多出现于：学校网络、路由模式下的光猫、家用路由器嵌套场景等



解决方案 (Solution)

- 让上级路由器支持 DHCPv6-PD。
- 没有了。

宽带运营商给的光猫不好用？
只能用路由模式，不给分配 DHCPv6-PD？
记好网络参数，去闲鱼再买个光猫！

校园网不给分配 DHCPv6-PD？
一■二■三■■，发■■■■，不
行再■■■■，去■■■■前面
■■■■，最后■■■■■■■■！



妥协方案 (Workaround)

- 非官方的 NAT6 方案。
- 6to4 隧道技术。
- 对于服务器：使用 socat 等进行端口转发。



非官方的 NAT6 方案

思路：

LAN 网段不再是 WAN 的一个子网，手动划出一个网段，如 IPv6 的未定义网段。不使用 IPv6 内网网段（fc 开头或 fd 开头），否则设备可能认为无 IPv6 互联网。dd 开头即为一段未定义网段，可使用它。

配置 ip6tables 以便启用非官方的 IPv6 地址转换，aka. NAT6。

OpenWRT 的 UPnP 协议已经支持 NAT6，因此仍然可使用 BT 等 P2P 工具。

理论上，依赖 IPv6 地址全球可访问的应用将无法正常使用，但目前还没有出现。

<https://www.right.com.cn/forum/thread-2661027-1-1.html>



socat 端口转发

应用场景：

山东大学镜像站，仅分配了一个 IPv6 地址，未分配网段，因此容器要么使用非官方的 NAT6 技术，要么通过端口转发的方式向外界提供服务，但容器不能直接连接到 IPv6 网络。目前使用后者，即容器只使用 IPv4 内网地址，配置端口转发。

使用 socat 程序将 IPv4 的 443 端口转发到本机 IPv6 地址的 443 端口：

```
/usr/bin/socat tcp6-listen:443,fork,reuseaddr,su=nobody TCP4:192.168.0.2:443
```

socat 这种用户态的端口转发对性能有一定的影响。但 Linux 内核不支持直接跨 IPv4/IPv6 协议转发，目前只能使用这种方式。



Thanks!

Any **questions** ?



齐划一



qi@huayi.email



山东大学