虚拟化

学习目标

内容

```
快速入门
    了解虚拟化是什么，有哪些分类。怎么实现
    kvm环境部署

核心知识
    kvm本身的各种核心资源的核心命令操作
        kvm它有多套命令集合
拓展知识
    kvm实现虚拟化的各个级别的原理解析
    kvm操作虚拟机的web界面(一个)
```

环境

```
1台ubuntu主机即可
    4g内存，cpu1-2 够用
    kvm虚拟机： centos8.4

软件：
    kvm本身的软件、libvirt、brctl-查看主机上的网络设备关联关系
```

目标

```
1 大家学习 虚拟化的相关知识(kvm)
2 挑战一下大家当天知识吸收的底线
```

回顾知识

```
bash 各种基础语法
    变量、表达式、逻辑控制
```

快速入门

虚拟化基础

小结

```
虚拟化
    通过技术手段，在当前的资源基础上，隔离出来几个独立的空间

技术方式：
    在硬件基础上进行虚拟化 - xen
    在系统基础上进行虚拟化 - kvm

实现方式：
    模拟        - 在操作系统上，安装软件，模拟大部分的功能，受底层系统限制
    全虚拟化    - 通过软件技术(辅助硬件技术) 实现完全独立的虚拟主机资源(VM自以为自己是真是的物理主机)
    半虚拟化    - 通过综合技术,实现部分功能的虚拟化，性能方面主要是直接以硬件实现
    硬件辅助虚拟化 - 直接借助于硬件的能力实现虚拟化功能-性能非常好(代价非常高)

目标：
    空间和时间上，提高工作效率
```

KVM基础

小结

```
kvm
    1 全虚拟化解决方案
    2 开源的(内核有限制 2.6.20，对于我们的ubuntu20.10来说，无所谓)

原理解析
    1 完成的主机系统
    2 内核加载kvm
    3 kvm外来的霸道总裁
    4 内核空间演变为 hypervisor
    5 虚拟机以 来宾模式(用户空间+内核空间) 进行存在

特点
    全虚拟化解决方案、开源
    多种存储方案，支持cpu、memery虚拟化、支持硬件辅助功能、等等
```

kvm安装

小结

```
软件前提
    vm workstation 开启 inter-vt| amd-v
    lsmod | grep kvm

软件安装
    apt install -y qemu-kvm libvirt-daemon-system  libvirt-
clients bridge-utils  virtinst  virt-manager virt-viewer
    注意:
        两套管理命令
            qemu-*
            virsh *
效果展示
    1 多了一个网桥 virtbr0  192.168.122.0
    2 工作目录 /etc/libvirt/
        qemu - 虚拟机专属目录，内涵 vm主机的资源配置和网络等
```

kvm命令解析

小结

```
两套管理命令
    qemu-*
        qemu-img 管理磁盘存储等
    virsh *
        virt-install 安装虚拟机
        virsh 管理虚拟机
```

核心知识

基础实践

初始化

小结

磁盘创建

```
qemu-img create -f raw /opt/CentOS-8-x86_64.raw 10G
```

系统创建

```
virt-install --virt-type=kvm --name=CentOS-8-x86_64 --memory 2048 --cdrom=/data/softs/CentOS-8.4.2105-x86_64-boot.iso --disk=/opt/CentOS-8-x86_64.raw  --network network=default --graphics vnc,listen=0.0.0.0 --noautoconsole
```

注意：

默认情况下，使用 tighvnc连接虚拟机的时候，用的端口是 5900

创建好的虚拟机，本质上就是操作系统的一个 进程

检查和创建

小结

创建好的虚拟机，配置文件在

/etc/libvirt/qemu/虚拟机.xml

/etc/libvirt/storage/存储.xml

虚拟机查看

virsh list --all|--persistent|--transistent|--autostart

默认的虚拟机是 持久类型的、不自动启动的

创建虚拟机

前提：要准备一个配置文件

virsh create

基于虚拟机配置文件临时启动一台虚拟机

- 一旦关闭了，就找不到了

virsh define + virsh start

基于虚拟机配置文件先注册到管理平台，然后再启动

- 虚拟机关闭后，还可以在管理平台看到

配置文件要点：暂时先更改两项：name 和 系统的uuit

```
<domain type='kvm'>
  <name>CentOS-8-x86_64-1</name>
  <uuid>5b2587ec-9774-4b60-b5ee-5dd0d74b88aa</uuid>
  <memory unit='KiB'>2097152</memory>
  <currentMemory unit='KiB'>2097152</currentMemory>
```

```
进入虚拟机
    方法1 ：
        virsh domdisplay 虚拟机名 -- 查看连接uri
        使用vnc 连接 uri地址
        -- 特点：慢，vnc里面的命令终端无法粘贴复制等操作
    方法2： kvm 提供了专用的命令行连接终端
        virsh console 虚拟机名称
        注意：默认情况下使用不成功，它依赖于虚拟机的对外终端配置
```

挂起和恢复

小结

```
挂起
    - 暂停虚拟机的运行
    virsh suspend 虚拟机名
恢复
    - 恢复虚拟机的运行
    virsh resume 虚拟机名
```

关闭和删除

小结

```
关闭
    正规关停虚拟机的运行
    virsh shutdown 虚拟机
    注意：
        因为涉及到虚拟设备的操作，依赖于 acpid服务(对于ubuntu来说，无所谓)

    强制关闭虚拟机
    virsh destroy 虚拟机
    注意：
        强制关闭一个正在运行的虚拟机，配置文件等不受影响。

    超级强制管理
    kill -9 $(ps aux | grep 虚拟机关键字 | grep -v 'grep' | awk
'{print $2}')
删除
    删除虚拟机
    virsh undefine 虚拟机
    注意：
        不仅仅将虚拟机从 管理平台里面清空，而且还把
            /etc/libvirt/quem/配置文件.xml 也删除掉
```

开机自启动

小结

```
定位：
    随着 kvm 服务(libvirtd)的开启，我们的虚拟主机自动启动
    开启：virsh autostart 虚拟机名
    关闭：virsh autostart --disable 虚拟机名
    查看：
        virsh list --autostart | --no-autostart 虚拟主机名
```

备份和编辑

小结

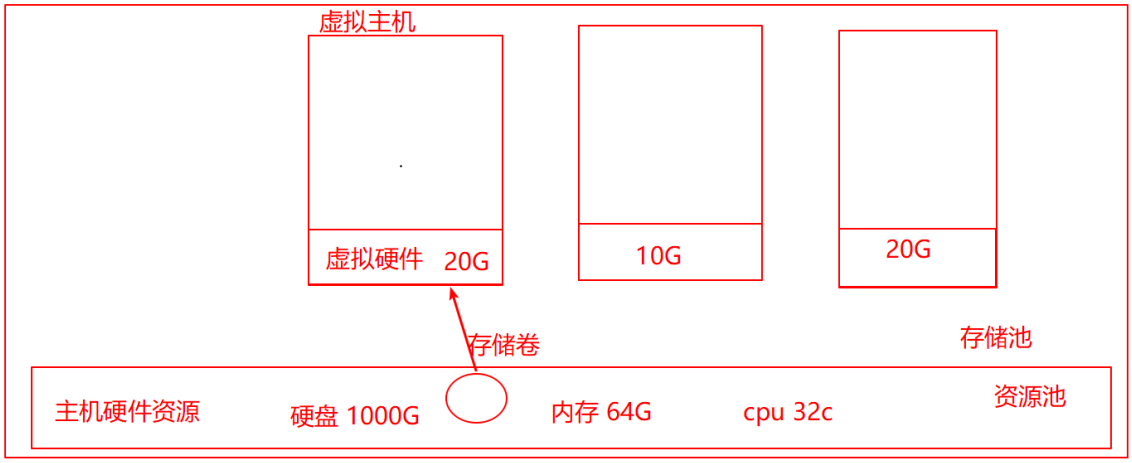
```
备份：
    virsh 专用的虚拟主机配置文件拷贝，类似于 cp
    virsh dumpxml 虚拟机名
编辑：
    virsh edit 虚拟机名
    注意：
        文件在进行编辑的时候，有可能会出现一些异常：
            1 修改内容失败
                -- 资源冲突导致
            2 某些权限限制导致，无法修改
        在使用virsh edit 编辑后的虚拟机文件进行某些复杂操作之前，必须检查一下，配置是否生效了。
            - 大家在 自定义网络 实践的时候，尤其要注意。
```

```
root@python-auto:/etc/libvirt/qemu# virsh edit CentOS-8-x86_64
error: operation failed: domain 'CentOS-8-x86_64' already exists with uuid 5b2587ec-9774-4bb0-b5ee-5
dd0d74b88aa
Failed. Try again? [y,n,i,f,?]:
```

存储管理

存储池基础

简介



存储池

是宿主机资源为了方便 各个虚拟主机的使用，将本地的所有资源放到一个 抽象的资源池里，这个资源池就是 存储池

存储卷

存储卷是方便 各个虚拟主机本身存储数据使用的文件，他是来源于 存储池里面的一个空间区域(抽象的)

使用逻辑

1 创建存储池
2 创建存储卷
3 挂载存储卷到虚拟主机
4 虚拟主机格式化存储卷
5 挂载存储卷到某个目录
6 文件系统的正常使用

命令简介

信息查看

```
virsh pool-list
    --transient
    --persistent
    --autostart
    --all
```

创建存储池

临时创建

```
virsh pool-create-as 存储池名 --type 类型 --target=目标
```

持久创建

```
virsh pool-define-as 存储池名 --type 类型 --target=目标
```

存储池基本操作

小结

```
开启
    普通开启
        virsh pool-start
    开机自启动
        virsh pool-autostart

关闭
    virsh pool-destroy

取消
    删除存储池目录
        virsh pool-delete
    删除存储池配置文件 -- 取消注册
        virsh pool-undefine
```

存储卷基础

小结

```
创建
    virsh vol-create-as 存储池 存储卷 --capacity 容量 --allocation 初始容量 --format 存储格式

信息查看
    virsh vol-key    存储卷 --pool 存储池
    virsh vol-path   存储卷 --pool 存储池
    virsh vol-info   存储卷 --pool 存储池
    virsh vol-list 存储池
    virsh vol-dumpxml    存储卷 --pool 存储池
```

存储卷实践

小结

挂载
    virsh attach-disk 虚拟机 数据卷 挂载名称
    注意：
        只有格式化后，才可以正常的给文件系统使用
卸载
    virsh detach-disk 虚拟机 挂载名称
    注意：
        只有先取消挂载，再卸载磁盘

存储卷在进行挂载的时候，必须保证虚拟机是开启的

```
root@python-auto:/etc/libvirt# virsh attach-disk CentOS-8-x86_64 /kvm/dpool/dpool-vol-1 vda
error: Failed to attach disk
error: Requested operation is not valid: domain is not running
```

数据实践

小结

内容导出
    virsh vol-download
内容导入
    virsh vol-upload
内容擦除
    virsh vol-wipe
容量调整
    virsh vol-resize
    注意：
        容量调整，只能调大，不能调小。

克隆删除

小结

数据卷克隆
    virsh vol-clone
数据卷删除
    virsh vol-delete

自学思路

```
1 virsh help | grep 资源类型
    - 对所有的命令有一个概况性的认知
2 对所有的命令进行简单的梳理
    创建、查看、删除、其他
3 针对的学习
    virsh 子命令 --help
        -- 看基本语法
```

网络管理

网络基础

小结

```
网络的流转顺序
    1 用户来了 - ubuntu(ens33) - if_forward(网络地址转换) - ubuntu(vnet0)
    2 ubuntu(vnet0) - kvm(virtbr0) - 虚拟机(eth0)

基本信息
    virsh net-list
    virsh net-info
    virsh net-dumpxml
    virsh net-autostart
```

基本实践

小结

```
创建网卡
    virsh net-create
    virsh net-define
关闭网卡
    virsh net-destroy
删除网卡
    virsh net-undefine
```

自定义网络

小结

```
kvm的网络模型：
    默认是 net模型，网络地址转换，量大的时候，导致资源消耗过多
```

我们希望使用桥接模式，虚拟机和宿主机在同一个网段，数据传递效率高

流程

1 创建专用的虚拟网桥

```
brctl addbr br0
brctl stp br0 on
```

2 配置虚拟网桥

```
ifconfig ens33 0 up
brctl addif br0 ens33
ifconfig br0 10.0.0.12/24 up
route add default gw 10.0.0.2
```

3 虚拟机使用虚拟网桥

```
<interface type='bridge'>
  <mac address='52:54:00:d0:e5:28'/>
  <source bridge='br0'/>
  <model type='virtio'/>
  。。。
</interface>
```

4 虚拟机内部定制网卡

```
[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
...
BOOTPROTO=static
...
IPADDR=10.0.0.122
NETMASK=255.255.255.0
GATAWAY=10.0.0.2
DNS1=10.0.0.2

nmcli c reload
nmcli c up eth0
nmcli d reapply eth0
nmcli d connect eth0
```

注意：

如果在ip能够正常ping通的话，说明整体配置就搞定了。

如果域名没有成功，问题在两个地方：

1 配置文件里面的 DNS1=xxx

2 dns的专属配置文件 /etc/resolv.conf

镜像管理

基础知识

小结

```
kvm支持大量的后端存储格式
    qemu-img --help

格式特点：
    raw     - 实际占用的空间很大，容量由设计时候来指定
    qcow2   - 存储的实际大小由真实的数据容量来决定，而不是设计的那个大小

常见命令
    信息查看
        qemu-img info

    创建磁盘
        qemu-img create -f 磁盘格式 磁盘名称 磁盘容量
    转换磁盘
        qemu-img convert -f 原格式 原文件 -o 转换后格式 转换后磁盘文件
```

克隆

小结

```
方式1： 虚拟机完整克隆
    virt-clone -o 现有虚拟机名称 -n 克隆后的虚拟机名称 -f 克隆后的虚拟机磁盘文件
方式2： 虚拟机定制克隆(作业)
    2-1 备份文件
        virsh dumpxml 原虚拟机 > 新虚拟机
    2-2 转换磁盘
        qemu-img -f 原格式 原磁盘文件 -o 新格式 新磁盘文件
    2-3 更改属性
        虚拟机名称、uuid、磁盘文件、网卡信息

进阶作业：
    定制网络(桥接)
        - 仅仅是将虚拟机的网络配置转换成 bridge 模式即可，虚拟机内部不用做任何操作

升级作业：
    定制新的net网卡名称(网段等)
    将新的虚拟机应用新的网段，并自动获取ip地址
```

```
升级作业2：
    将dhcp方式改为静态方式

超级作业：
    跨网段的主机通信
        route add default gw xxx
```

资源管理

cpu资源

小结

```
简介
    kvm 支持CPU的虚拟化，而且是热调整(在不断电的时候，调整vm主机的cpu数量)
命令：
    信息查看
        virsh dominfo
        virsh cpu-stats
        virsh vcpuinfo          查看虚拟机cpu的具体分布和插槽亲和性
        virsh vcpucount         仅仅查看虚拟机的cpu的数量信息(最大和最小|配置和运行)
    CPU调整
        virsh setvcpus
            --config            调整配置文件里面的当前cpu的数量
            --live              调整当前虚拟机运行受的cpu数量
    cpu绑定
        virsh vcpupin 虚拟机名 vcpu编号 cpu列表位置
            主机少的情况下，没有必要调整这个
```

内存实践

小结

```
简介
    我们可以对虚拟机的运行内存进行热调整
    注意：
        如果我们的虚拟机在运行某种应用，不要乱动内存，动的话，你试试

命令
    方法1
        virsh qemu-monitor-command 虚拟机名 --hmp 子命令 操作
            命令信息查看：info
            命令帮助：help [子命令]
```

内存调整：balloon、mem、

方法2

```
virsh setmem
virsh setmaxmem
virsh dommemstat
virsh dompxml 虚拟机 | grep mem
virsh dominfo 虚拟机 | grep mem
```

磁盘实践

小结

简介

我们可以通过 qemu-img 对系统的数据存储磁盘进行各种操作

注意：

不要对包含操作系统的磁盘进行各种xx

命令

磁盘的操作

信息查看 qemu-img info

磁盘创建 qemu-img create -f 格式 名称 大小

磁盘调整 qemu-img resize 磁盘名称 +|- 大小

注意：

磁盘大小转换的时候，会自动调整磁盘的一些元数据信息，比如格式等

磁盘的挂载和卸载

```
virsh attach-disk 虚拟机 磁盘 目标名称 参数
virsh detach-disk 虚拟机 目标名称 参数
```

虚拟化原理

cpu虚拟化

小结

简介

cpu 本身就实现了一种类似虚拟化的效果，这个效果是基于cpu时间分片的机制来实现的

CPU 在进行各种操作指令执行的时候，会涉及到两种情况：

普通的文件系统级别的操作命令

- 在用户空间执行
- ring3

涉及到底层硬件几倍的操作命令

- 命令在用户空间，但是执行需要内核空间来执行
- 涉及到系统调用
- ring0

虚拟分类
模拟
- 软件实现
- 用户空间模拟、内核空间
- 双层系统调用
vm内部的用户空间和内核空间 -- 本身就是 宿主机的用户空间
宿主机的用户空间和内核空间
全虚拟化
- 软件实现，但是vm自己不知道
- BT直接将vm用户空间的命令进行转换到宿主机内核空间
半虚拟化
- 用户空间软件实现 - ring1-3
- 内核空间宿主机实现 - 直接用宿主机的ring0
硬件辅助虚拟化
- 直接在硬件的基础上，整体划分一块独立的区域，与 宿主机的 ring一一对应

内存虚拟化

小结

简介
我们在进行程序开发的时候，涉及到内存的虚拟化功能
逻辑地址 - 线性地址
物理地址
这两个地址是通过 关联 一一对应的
隐藏：
为了加快数据的快速操作，需要 缓存下来对应关系
参考资料 -- https://baike.baidu.com/item/MMU/4542218

虚拟化
模拟
将我们的 vm主机的逻辑地址和物理地址的映射，再转换一层
mmu虚拟
直接将vm主机的寻址另开一条线，直接关联到 真实是宿主机的内存
原来的 映射关系依然保持--程序的需要
进阶管理
将vm主机身的内存 进行更高一级别的内存高效管理 -- 物理的内存

网络虚拟化

小结

```
注意：
    因为在进行网络数据传递的时候，涉及到底层网卡之类设备的操作，
        可以参考在cpu虚拟化中介绍的相关信息，设备io虚拟化的相关信息

模式
    默认 - nat 网络地址转换
        原理(用户访问虚拟机的角度)
            用户数据-宿主机网卡ens33 -- 内核参数(ip_forward) - 虚拟网卡(vnet0) - vm网桥(virtbr0) - vm网卡(eth0)

    桥接 - bridge
        原理：(用户访问虚拟机的角度)
            用户数据-宿主机网卡ens33(虚拟网桥br0有ip地址) -- vm主机网卡(eth0)
            网络线
                用户数据-虚拟网桥br0 -- vm主机网卡(eth0)
            数据线
                用户数据-宿主机网卡ens33 -- vm主机网卡(eth0)
    主机 - host
        原理：vm主机直接使用宿主机的网络信息
            用户数据 - 宿主机网卡(直接交给vm主机)
    隔离
        原理：创建后的虚拟主机，使用独有的网络环境，可以实现高度的定制化
```

io虚拟化

小结

```
设备的基本操作
    1 用户在用户空间进行程序的开发
    2 通过系统调用，借助于设备驱动，操作底层设备
    3 底层设备执行操作

io虚拟化(以网卡设备为例)
    模拟
        vm主机内部 走一个标准的设备系统调用  -- 本质上还是宿主机用户空间的操作
        宿主机内部 走一个标准的设备系统调用
    半虚拟化
        将模拟的系统调用精简，以一个硬件的驱动来搞定
```

这个硬件驱动划分为了两部分：
前端驱动 - 给vm主机用
后端驱动 - 给宿主机用

透传
借助于专用的管理机制(hypervisor管理的设备驱动)，由vm主机直接将数据交给底层设备

硬件虚拟化
将所有的的底层硬件统一管理，然后vm主机借助于DMA来操作底层设备

注意:

每隔虚拟化里面都会涉及到大量的汉语拼音简写，自己去梳理。

批量管理

小结

```
kvm手工操作繁琐，各种命令多，通过web界面精简操作步骤，提高工作效率
    web界面很多：
        wok+kimchi  -- kvm插件的方式来存在的
            需要定制化安装 -- 因为有些东西需要底层操作系统环境的支持
        virt-manager -- kvm自带的管理工具
            简单、高效，默认情况下，只能以普通用户来进行启动

wok+kimchi 配置
    1 基本软件环境的安装
    2 wok软件的安装
        可以参考解压包里面文件的 README.txt 内容
            -- 注意系统版本的区别
    3 kimchi软件的安装
    4 apache2的环境配置
        模块加载+配置文件定制
```

```
sudo apt install -y python3-configobj python3-lxml python3-magic
python3-paramiko python3-ldap spice-html5 novnc qemu-kvm python3-
libvirt python3-parted python3-guestfs python3-pil python3-
cherrypy3 libvirt0 libvirt-daemon-system libvirt-clients nfs-
common sosreport open-iscsi libguestfs-tools libnl-route-3-dev
python3-cheetah gcc make autoconf automake git python3-pip
python3-requests python3-mock gettext pkgconf xsltproc python3-
dev pep8 pyflakes python3-yaml apache2

sudo apt-get remove python3-cherrypy3
sudo pip3 install cherrypy ethtool python-pam psutil jsonschema
```