

本站文章大部分为作者原创，非商业用途转载无需作者授权，但务必在文章标题下面注明作者 刘世民 (Sammy Liu) 以及可点击的本博客地址
 超级链接 <http://www.cnblogs.com/sammyliu/>，谢谢合作



世民谈云计算

(声明：本站文章皆基于公开来源信息，仅代表作者个人观点，与作者所在公司无关)

昵称：SammyLiu
园龄：2年6个月
荣誉：推荐博客
粉丝：470
关注：30
[+加关注](#)

[博客园](#) [首页](#) [新随笔](#) [订阅](#) [XML](#) [管理](#)

随笔-121 评论-504 文章-45

2017年5月						
日	一	二	三	四	五	六
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

常用链接

- [我的随笔](#)
- [我的评论](#)
- [我的参与](#)
- [最新评论](#)
- [我的标签](#)

我的标签

- GRE(1)
- Neutron(1)
- Open vSwitch(1)
- OpenStack(1)

随笔分类(254)

- Ceilometer(3)
- Ceph(13)
- Cinder(6)
- Docker(8)
- Glance
- Heat(2)
- K8S
- Keystone(1)
- KVM(10)
- MessageQueue(4)
- MySQL(1)
- Neutron(17)
- Nova(10)
- OpenStack(33)
- Sahara
- Storage(1)
- Swift(3)
- Trove

KVM 介绍 (5) : libvirt 介绍 [Libvirt for KVM/QEMU]

学习 KVM 的系列文章：

- (1) 介绍和安装
- (2) CPU 和 内存虚拟化
- (3) I/O QEMU 全虚拟化和准虚拟化 (Para-virtualization)
- (4) I/O PCI/PCIe设备直接分配和 SR-IOV
- (5) libvirt 介绍
- (6) Nova 通过 libvirt 管理 QEMU/KVM 虚拟机
- (7) 快照 (snapshot)
- (8) 迁移 (migration)

1. Libvirt 是什么

为什么需要Libvirt?

1. Hypervisor 比如 qemu-kvm 的命令行虚拟机管理工具参数众多，难于使用。
2. Hypervisor 种类众多，没有统一的编程接口来管理它们，这对云环境来说非常重要。
3. 没有统一的方式来方便地定义虚拟机相关的各种可管理对象。

Libvirt提供了什么？

1. 它提供统一、稳定、开放的源代码的应用程序接口 (API)、守护进程 (libvirtd) 和和一个默认命令行管理工具 (virsh)。
2. 它提供了对虚拟化客户机和它的虚拟化设备、网络和存储的管理。
3. 它提供了一套较为稳定的C语言应用程序接口。目前, 在其他一些流行的编程语言中也提供了对libvirt的绑定, 在Python、Perl、Java、Ruby、PHP、OCaml等高级编程语言中已经有libvirt的程序库可以直接使用。
4. 它对多种不同的 Hypervisor 的支持是通过一种基于驱动程序的架构来实现的。libvirt 对不同的 Hypervisor 提供了不同的驱动, 包括 Xen 的驱动, 对 QEMU/KVM 有 QEMU 驱动, VMware 驱动等。在 libvirt 源代码中, 可以很容易找到 qemu_driver.c、xen_driver.c、xenapi_driver.c、vmware_driver.c、vbox_driver.c 这样的驱动程序源代码文件。
5. 它作为中间适配层, 让底层 Hypervisor 对上层用户空间的管理工具是可以做到完全透明的, 因为 libvirt 屏蔽了底层各种 Hypervisor 的细节, 为上层管理工具提供了一个统一的、较稳定的接口 (API)。
6. 它使用 XML 来定义各种虚拟机相关的受管理对象。

目前，libvirt 已经成为使用最为广泛的对各种虚拟机进行管理的工具 and 应用程序接口（API），而且一些常用的虚拟机管理工具（如virsh、virt-install、virt-manager等）和云计算框架平台（如OpenStack、OpenNebula、Eucalyptus等）都在底层使用libvirt的应用程序接口。

Ubuntu(3)

VMware(3)

安装和配置(1)

版本(4)

备份(1)

大数据(5)

翻译(4)

高可用 (HA) (6)

基础知识(19)

监控(1)

容器(4)

容器编排

使用案例(4)

网络(8)

问题定位(3)

行业(14)

性能(4)

虚拟化(7)

原理(22)

云Cloud(29)

随笔档案(121)

2017年5月 (1)

2017年3月 (1)

2017年1月 (1)

2016年10月 (7)

2016年9月 (5)

2016年8月 (4)

2016年7月 (1)

2016年6月 (5)

2016年5月 (1)

2016年4月 (1)

2016年3月 (9)

2016年2月 (4)

2016年1月 (2)

2015年12月 (7)

2015年11月 (7)

2015年10月 (4)

2015年9月 (4)

2015年8月 (5)

2015年7月 (9)

2015年6月 (10)

2015年5月 (3)

2015年4月 (11)

2015年3月 (2)

2015年2月 (6)

2015年1月 (5)

2014年12月 (6)

文章分类(21)

Ceph(1)

Application layer

libvirt tools layer

libvirtd

virsh

API bindings

libvirt API layer

Hypervisor layer

Other hypervisor drivers

libxen

CIM XML

Identity

DC Automation

Virtualization Mgmt

Hyp Mgmt

Virt HW

Hypervisor

OpenLDAP+Kerberos

SLE-HAE

sudo

Policykit

vm-install

virsh

virt-man

virt-viewer

libvirt-cim

libvirtd

libvirt

...

Xen

xenlight

QEmu-KVM

...

xm

xl

xend

libxenlight

qemu-kvm

xen bus

qemu-dm

virtio

Xen

KVM

(SLE 11)

1.1 Libvirt C API

1.1.1 Libvirt API 所管理的主要对象

对象	解释
Domain (域)	指运行在由Hypervisor提供的虚拟机器上的一个操作系统实例 (常常是指一个虚拟机) 或者用来启动虚机的配置。
Hypervisor	一个虚拟化主机的软件层
Node (主机)	一台物理服务器。
Storage pool (存储池)	一组存储媒介的集合, 比如物理硬盘驱动器。一个存储池被划分为小的容器称作卷。卷会被分给一个或者多个虚机。
Volume (卷)	一个从存储池分配的存储空间。一个卷会被分给一个或者多个域, 常常成为域里的虚拟硬盘。

GlusterFS

Web 服务器(2)

操作系统(1)

存储

大数据(2)

分布式系统

服务器(1)

网络(11)

虚拟化(3)

云

文章档案(42)

2016年10月 (2)

2016年9月 (1)

2016年6月 (1)

2016年5月 (3)

2015年12月 (4)

2015年10月 (5)

2015年9月 (2)

2015年6月 (1)

2015年4月 (23)

积分与排名

积分 - 286831

排名 - 535

最新评论

1. Re:Neutron 理解 (1): Neutron 所实现的虚拟化网络 [How Netruon Virtualizes Network]
eth1 - 公共网络 (untagged) , 管理网络 (tag=102) , 存储网络 (tag=103) 不好意思, 大家共用同一个eth1端口的时候, 请问这里交换机端口是配置为tagged还是untagged.....
--xianke9

2. Re:理解Docker (5) : Docker 网络
1.12版本上网络的表现如何?
--幽灵狼

3. Re:理解Docker (5) : Docker 网络
我想请问一下运行docker quickstart terminal时一直卡在"waiting for an IP"应该如何解决呢? 希望楼主能解答一下。
--silentbell

4. Re:理解Docker (6) : 若干企业生产环境中的容器网络方案
写得好! 加油。
--itbj00

5. Re:理解Docker (5) : Docker 网络
非常好, 写得很详细。加油!
--itbj00

阅读排行榜

1.1.2 对象的管理模型

对象名称	对象	Python 类	描述
Connect	与 Hypervisor 的连接	virConnectPtr	在调用任何 API 去管理一个本地或者远端的Hypervisor前, 必须建立和这个Hypervisor的连接。
Domain	虚拟机	virDomainPtr	用于列举和管理已有的虚机, 或者创建新的虚机。唯一标识: ID,Name , UUID。一个域可能是暂时性的或者持久性的。暂时性的域只能在它运行期间被管理。持久性的域在主机上保存了它的配置。
VirtualNetwork	虚拟网络	virNetworKPtr	用于管理虚机的网络设备。唯一标识: Name , UUID。一个虚拟网络可能是暂时性的或者持久性的。每个主机上安装libvirt后, 它都有一个默认的网络设备"default"。它向该主机上运行的虚机提供DHCP服务, 以及通过NAT连接到主机上。
StoragePool	存储池	virStoragePoolPtr	用于管理虚拟机内的所有存储, 包括 local disk, logical volume group, iSCSI target, FibreChannel HBA and local/network file system。唯一标识: Name , UUID。一个存储池可能是暂时性的或者持久性的。Pool 的 type 可以是 dir, fs, netfs, disk, iscsi, logical, scsi,mpath, rbd, sheepdog , gluster 或者 zfs。
StorageVolume	存储卷	virStorageVolPtr	用于管理一个存储池内的存储块, 包括一个池内分配的块、磁盘分区、逻辑卷、SCSI/iSCSI Lun, 或者一个本地或者网络文件系统内的文件等。唯一标识: Name , Key , Path。
HostDevice	主机设备	virNodeDevPtr	用于管理主机上的物理硬件设备, 包括 the physical USB or PCI devices and logical devices these provide, such as a NIC, disk, disk controller, sound card, etc。唯一标识: Name。

1.1.3 API 的简单分类

Libvirt API 就是对各种对象的各种操作, 包括基本的增、删、改、查操作和其它操作。

对象	增	删	改	查	其它
Connect	virConnectOpen virConnectOpenAuth virConnectOpenReadOnly	virConnectClose	virConnectSetKeepAlive		
Storage	virStorageP	virStorageP	virStoragePoolRefresh	virConnectFindStor	virStorage

1. Neutron 理解 (1): Neutron 所实现的虚拟化网络 [How Netruon Virtualizes Network](22087)
2. 理解 OpenStack 高可用 (HA) (1) : OpenStack 高可用和灾备方案 [OpenStack HA and DR](13707)
3. Neutron 理解 (3): Open vSwitch + GRE/VxLAN 组网 [Netruon Open vSwitch + GRE/VxLAN Virutal Network](13434)
4. 探索 OpenStack 之 (9) : 深入块存储服务Cinder (功能篇) (12921)
5. 理解 OpenStack + Ceph (1) : Ceph + OpenStack 集群部署和配置 (12444)

评论排行榜

1. Neutron 理解 (1): Neutron 所实现的虚拟化网络 [How Netruon Virtualizes Network](63)
2. Neutron 理解 (14) : Neutron ML2 + Linux bridge + VxLAN 组网 (54)
3. Neutron 理解 (8): Neutron 是如何实现虚拟机防火墙的 [How Neutron Implements Security Group](34)
4. Neutron 理解 (3): Open vSwitch + GRE/VxLAN 组网 [Netruon Open vSwitch + GRE/VxLAN Virutal Network](25)
5. Neutron 理解 (5) : Neutron 是如何向 Nova 虚拟机分配固定IP地址的 (How Neutron Allocates Fixed IPs to Nova Instance) (21)

推荐排行榜

1. Neutron 理解 (1): Neutron 所实现的虚拟化网络 [How Netruon Virtualizes Network](9)
2. 我所了解的 京东、携程、eBay、小米 的 OpenStack 云(6)
3. 理解 OpenStack 高可用 (HA) (1) : OpenStack 高可用和灾备方案 [OpenStack HA and DR](6)
4. Neutron 理解 (2): 使用 Open vSwitch + VLAN 组网 [Netruon Open vSwitch + VLAN Virutal Network](6)
5. 理解 OpenStack 高可用 (HA) (2) : Neutron L3 Agent HA 之 虚拟路由冗余协议 (VRRP) (5)

ora ge po ol	oolBuild virStorageP oolCreate virStorageP oolCreateX ML virStorageP oolDefineX ML	oolDelete virStorageP oolDestroy virStorageP oolFree virStorageP oolUndefine	virStoragePoolSetAutostar t	agePoolSources virConnectListAllSt oragePools virConnectListDefin edStoragePools virConnectListStora gePools virConnectNumOfD efinedStoragePools virConnectNumOfS toragePools virStoragePoolGetI nfo/Name/UUID/U UIDString/XMLDes c virStoragePoolsA ctive/Persistent virStoragePoolLook upByName/UUID/U UIDString/Volume virStoragePoolRef	PoolGetA utostart virStorage PoolGetC onnect virStorage PoolNum OfVolume s virStorage PoolListAl lVolumes virStorage PoolListV olumes
Sto rag e vol um e	virStorageV olCreateXML virStorageV olCreateXML LFrom	virStorageV olDelete virStorageV olFree	virStorageVolResize virStorageVolUpload virStorageVolWipe virStorageVolWipePattern	virStorageVolGetC onnect/Info/Key/Na me/Path/XMLDesc virStorageVolLook upByKey/Name/Pat h virStorageVolRef	virStorage VolDownl oad
Net wo rk	virNetwork Create virNetwork CreateXML virNetwork DefineXML	virNetwork Destroy virNetworkF ree virNetwork Undefine	virNetworkSetAutostart virNetworkUpdate	virConnectListAllNe tworks virConnectListDefin edNetworks virConnectListNetw orks virConnectNumOfD efinedNetworks virConnectNumOfN etworks virNetworkGetBridg eName/DHCPLeas es/Name/UUID/UI DString/XMLDesc virNetworkIsActive/ Persistent virNetworkLookup ByName/UUID/UI DString virNetworkGetAuto start virNetworkGetConn ect	virConnec tNetwork EventDer egisterAn y virConnec tNetwork EventGen ericCallba ck virNetwor kDHCPLe aseFree
Do ma in sn ap sh ot	virDomainS napshotCre ateXML	virDomainS napshotDel ete virDomainS napshotFre e	virDomainRevertToSnaps hot	virDomainHasCurr entSnapshot virDomainListAllSn apshots virDomainSnapshot Current virDomainSnapshot GetConnect/Domai n/Name/Parent/XM LDesc virDomainSnapshot	

					HasMetadata virDomainSnapshotIsCurrent virDomainSnapshotListAllChildren virDomainSnapshotListChildrenNames virDomainSnapshotListNames virDomainSnapshotLookupByName virDomainSnapshotNum virDomainSnapshotNumChildren virDomainSnapshotRef
	Host		virInitialize virNodeSetMemoryParameters virNodeSuspendForDuration	virConnectBaselineCPU virConnectCompareCPU virConnectGetCPUModelNames/Capabilities/Hostname/LibVersion/MaxVcpus/Sysinfo/Type/URI/Version virConnectIsAlive virConnectIsEncrypted virConnectIsSecure virGetVersion virNodeGetCPUMap/CPUStats /CellsFreeMemory/FreeMemory/Info/MemoryParameters/MemoryStats / virNodeGetSecurityModel	virTypedParamsAddBoolean virTypedParamsAddDouble virTypedParamsAddFromString virTypedParamsAddInt virTypedParamsAddLong virTypedParamsAddString virTypedParamsAddUInt virTypedParamsAddULLong virTypedParamsClear virTypedParamsFree virTypedParamsGet
	Interface ac e	virInterfaceCreate virInterfaceDefineXML	virInterfaceDestroy virInterfaceFree virInterfaceUndefine	virInterfaceChangeBegin virInterfaceChangeCommit virInterfaceChangeRollback	virConnectListAllInterfaces virConnectListDefinedInterfaces virConnectListInterfaces virConnectNumOfDefinedInterfaces virConnectNumOfInterfaces virInterfaceGetConnect virInterfaceGetMACString virInterfaceGetName virInterfaceGetXMLDesc virInterfaceIsActive virInterfaceLookupByMACString virInterfaceLookupByName virInterfaceRef

	Net Filter	virNWFilter DefineXML	virNWFilter Free virNWFilter Undefine	virConnectListAllNWFilters virConnectListNWFilters virConnectNumOfNWFilters virNWFilterGetName virNWFilterGetUUID virNWFilterGetUUIDString virNWFilterGetXMLDesc virNWFilterLookupByName virNWFilterLookupByUUID virNWFilterLookupByUUIDString virNWFilterRef		
	DomainEvent	virConnectDomainEventDeregister virConnectDomainEventDeregisterAny virConnectDomainEventDeviceAddedCallback virConnectDomainEventDeviceRemovedCallback	virConnectDomainEventAgentLifecycleCallback virConnectDomainEventBalloonChangeCallback virConnectDomainEventBlockJobCallback virConnectDomainEventCallback virConnectDomainEventDeviceRemovedCallback virConnectDomainEventDiskChangeCallback virConnectDomainEventGenericCallback virConnectDomainEventGraphicsCallback virConnectDomainEventIOErrorCallback virConnectDomainEventIOErrorReasonCallback virConnectDomainEventPMSuspendCallback virConnectDomainEventPMSuspendDiskCallback			
	Do	virDomainC	virDomainD	virDomainAbortJob	virConnectGetAllID	virConnect

main	create	destroy	virDomainAddIOThread	mainStats	tdomainXMLFromNative
	virDomainCreateLinux	virDomainDestroyFlags	virDomainDelIOThread	virConnectGetDomainCapabilities	tdomainXMLToNative
virDomainCreateWithFiles	virDomainFree	virDomainFree	virDomainAttachDeviceFlags	virConnectListAllDomains	tdomainXMLToNative
	virDomainCreateWithFlags	virDomainUndefine	virDomainDetachDeviceFlags	virConnectListDefinedDomains	tdomainXMLToNative
virDomainCreateXML	virDomainUndefineFlag	virDomainUndefineFlag	virDomainBlockCommit	virConnectNumOfDefinedDomains	
	virDomainCreateXMLWithFiles	virDomainUpdateDeviceFlags	virDomainBlockCopy	virConnectNumOfDomains	
virDomainDefineXML			virDomainBlockJobAbort	virDomainBlockStats	
	virDomainDefineXMLFlags		virDomainBlockJobSetSpeed	virDomainBlockStatsFlags	
virDomainDefineXMLFlags			virDomainBlockPeek	virDomainGetAutostart	
			virDomainBlockPull	virDomainGetBlkioParameters	
virDomainDefineXMLFlags			virDomainBlockRebase	virDomainGetBlockInfo	
			virDomainBlockResize	virDomainGetBlockIoTune	
virDomainDefineXMLFlags			virDomainCoreDump	virDomainGetBlockJobInfo	
			virDomainCoreDumpWithFormat	virDomainGetCPUStats	
virDomainDefineXMLFlags			virDomainFSFreeze	virDomainGetConnect	
			virDomainFSInfoFree	virDomainGetControlInfo	
virDomainDefineXMLFlags			virDomainFSThaw	virDomainGetDiskErrors	
			virDomainFSTrim	virDomainGetEmulatorPinInfo	
virDomainDefineXMLFlags			virDomainInjectNMI	virDomainGetFSInfo	
			virDomainInterfaceFree	virDomainGetHostName	
virDomainDefineXMLFlags			virDomainManagedSave	virDomainGetID	
			virDomainManagedSaveRemove	virDomainGetIOThreadInfo	
virDomainDefineXMLFlags			virDomainMigrate	virDomainGetInfo	
			virDomainMigrate2	virDomainGetInterfaceParameters	
virDomainDefineXMLFlags			virDomainMigrate3	virDomainGetJobInfo	
			virDomainMigrateSetCompressionCache	virDomainGetJobStats	
virDomainDefineXMLFlags			virDomainMigrateSetMaxDowntime	virDomainGetMaxMemory	
			virDomainMigrateSetMaxSpeed	virDomainGetMaxVcpus	
virDomainDefineXMLFlags			virDomainMigrateToURI	virDomainGetMemoryParameters	
			virDomainMigrateToURI2	virDomainGetMetadata	
virDomainDefineXMLFlags			virDomainMigrateToURI3	virDomainGetName	
			virDomainOpenChannel		
virDomainDefineXMLFlags			virDomainOpenConsole		
			virDomainOpenGraphics		
virDomainDefineXMLFlags			virDomainOpenGraphicsFD		
			virDomainPMSuspendForDuration		
virDomainDefineXMLFlags			virDomainPMWakeup		
			virDomainReboot		
virDomainDefineXMLFlags			virDomainReset		
			virDomainRestore		
virDomainDefineXMLFlags			virDomainRestoreFlags		
			virDomainResume		
virDomainDefineXMLFlags			virDomainSave		
			virDomainSaveFlags		

virDomainSaveImageDefineXML	virDomainGetNumaParameters
virDomainScreenshot	virDomainGetOSType
virDomainSendKey	
virDomainSendProcessSignal	virDomainGetSchedulerParameters
virDomainShutdown	virDomainGetSchedulerParametersFlags
virDomainShutdownFlags	
virDomainSuspend	virDomainGetSchedulerType
virDomainSetAutostart	
virDomainSetBlkioParameters	virDomainGetSecurityLabel
virDomainSetBlockioTune	virDomainGetSecurityLabelList
virDomainSetInterfaceParameters	virDomainGetState
virDomainSetMaxMemory	virDomainGetTime
virDomainSetMemory	virDomainGetUUID
virDomainSetMemoryFlags	virDomainGetUUIDString
virDomainSetMemoryParameters	virDomainGetVcpuPinInfo
virDomainSetMemoryStatsPeriod	virDomainGetVcpus
virDomainSetMetadata	virDomainGetVcpusFlags
virDomainSetNumaParameters	virDomainGetXMLDesc
virDomainSetSchedulerParameters	virDomainHasManagedSaveImage
virDomainSetSchedulerParametersFlags	virDomainIOThreadInfoFree
virDomainSetTime	
virDomainSetUserPassword	virDomainInterfaceAddresses
virDomainSetVcpus	virDomainInterfaceStats
virDomainSetVcpusFlags	
	virDomainIsActive
	virDomainIsPersistent
	virDomainIsUpdated
	virDomainListGetStats
	virDomainLookupByID
	virDomainLookupByName
	virDomainLookupByUUID
	virDomainLookupByUUIDString
	virDomainMemoryPeek
	virDomainMemoryStats
	virDomainMigrateGetCompressionCache
	virDomainMigrateG

				etMaxSpeed virDomainPinEmulator ? virDomainPinIOThread virDomainPinVcpu virDomainPinVcpuFlags virDomainSaveImageGetXMLDesc virDomainStatsRecordListFree	
Secret	virSecretDefineXML	virSecretFree virSecretUndefined	virSecretSetValue	virConnectListAllSecrets virConnectListSecrets virConnectNumOfSecrets virSecretGetConnect/UUID/UUIDString/UsageID/UsageType/Value/XMLDesc virSecretLookupByUUID/UUIDString/Usage virSecretRef	
Stream	virStreamNew	virStreamFree	virStreamFinish virStreamAbort virStreamRecv virStreamRecvAll virStreamSend virStreamSendAll	virStreamSinkFunc virStreamSourceFunc	

1.2 Libvirt XML 定义

Libvirt 使用 XML 来定义各种对象，其中，与 OpenStack Nova 关系比较密切的有：


disk (磁盘)	<p>任何磁盘设备，包括软盘 (floppy)、硬盘 (hard disk)、光驱 (cdrom) 或者半虚拟化驱动都使用 <disk> 元素来定义。方式： <disk type='*' device='*'>。其中：</p> <ul style="list-style-type: none">"type" 用来指定 device source 的类型："file", "block", "dir", "network", 或者 "volume"。具体的 source 由 <source> 标签定义。"device" 用来指定 device target 的类型："floppy", "disk", "cdrom", and "lun", 默认为 "disk"。具体的 target 由 <target> 标签定义。 <p>(1) "volume" 类型的 disk</p> <pre><disk type='volume' device='disk'> <driver name='qemu' type='raw'/> <source pool='blk-pool0' volume='blk-pool0-vol0' /> <target dev='hdk' bus='ide' /> </disk></pre> <p>(2) "file" 类型的 disk</p> <pre><disk type='file' snapshot='external'> <driver name="tap" type="aio" cache="default"/></pre>
-------------	---

```
<source file='/var/lib/xen/images/fv0'
startupPolicy='optional' />
<target dev='hda' bus='ide' />
</disk>
```


(3) "block" 类型的 disk

```
<disk type='block' device='cdrom'>
<driver name='qemu' type='raw' />
<target dev='hdd' bus='ide' tray='open' />
<readonly />
</disk>
```

(4) "network" 类型的 disk



```
<disk type='network' device='cdrom'>
<driver name='qemu' type='raw' />
<source protocol='http' name='url_path'>
  <host name='hostname' port='80' />
</source>
<target dev='hde' bus='ide' tray='open' />
<readonly />
</disk>
```



Host device
assignment (主
机设备分配)



```
<hostdev mode='subsystem' type='usb'> #USB 设备
直接分配
  <source startupPolicy='optional'>
    <vendor id='0x1234' />
    <product id='0xbeef' />
  </source>
  <boot order='2' />
</hostdev>
<hostdev mode='subsystem' type='pci'
managed='yes'> #PCI 设备直接分配
  <source>
    <address domain='0x0000' bus='0x06'
slot='0x02' function='0x0' />
  </source>
  <boot order='1' />
  <rom bar='on' file='/etc/fake/boot.bin' />
</hostdev>
```



Network interface
(网卡)

有几种 interface 类型 :

(1) type = 'network' 定义一个连接 Virtual network 的 interface



```
<devices>
  <interface type='network'>
    <source network='default' /> #虚拟网络的名称为
'default'
  </interface>
  ...
  <interface type='network'>
    <source network='default'
portgroup='engineering' />
    <target dev='vnet7' />
    <mac address='00:11:22:33:44:55' />
    <virtualport>
```

```
<parameters instanceid='09b11c53-8b5c-4eeb-8f00-d84eaa0aaa4f' />
</virtualport>
```

```
</interface>
</devices>
```



```
#virsh: attach-interface --domain d-2 --type network
--source isolatednet1 --mac 52:53:00:4b:75:6f --
config
```

(2) type='bridge' 定义一个 Bridge to LAN (桥接到物理网络) 的 interface: 前提是主机上存在一个 bridge, 该 bridge 已经连接到物理LAN。



```
<interface type='bridge'> #连接到 br0
  <source bridge='br0' />
</interface>
<interface type='bridge'> #连接到br1
  <source bridge='br1' />
  <target dev='vnet7' />
  <mac address='00:11:22:33:44:55' />
</interface>
<interface type='bridge'> #连接到 Open vSwitch
  bridge ovsbr
  <source bridge='ovsbr' />
  <virtualport type='openvswitch'>
    <parameters profileid='menial'
interfaceid='09b11c53-8b5c-4eeb-8f00-
d84eaa0aaa4f' />
  </virtualport>
</interface>
```



```
#virsh: attach-interface --domain d-2 --type bridge
--source virbr0 --mac 52:22:33:44:55:66 --config
```

(3) type='ethernet' 定义一个使用指定脚本连接到 LAN 的 interface

```
<devices>
  <interface type='ethernet'>
    <target dev='vnet7' />
    <script path='/etc/qemu-ifup-mynet' />
  </interface>
</devices>
```

(4) type='direct' 定义一个直接连接到物理网卡 (Direct attachment to physical interface) 的 interface: 需要 Linux macvtap 驱动支持

```
<interface type='direct'
trustGuestRxFilters='no'>
  <source dev='eth0' mode='vepa' />
</interface>
```

(5) type='hostdev' 定义一个由主机PCI 网卡直接分配 (PCI Passthrough) 的 interface: 分配主机上的网卡给虚拟机



```
<devices>
  <interface type='hostdev' managed='yes'>
    <driver name='vfio' />
    <source>
      <address type='pci' domain='0x0000'
```

```

bus='0x00' slot='0x07' function='0x0' />
</source>
<mac address='52:54:00:6d:90:02' />
<virtualport type='802.1Qbh'>
  <parameters profileid='finance' />
</virtualport>
</interface>
</devices>

```



network (网络)

```

<bridge name="virbr0" stp="on" delay="5"
macTableManager="libvirt" />
<domain name="example.com" localOnly="no" />
<forward mode="nat" dev="eth0" />

```

1. bridge : 定义一个用于构造该虚拟网络的网桥。
2. domain : 定义 DHCP server 的 DNS domain.
3. forward : 定义虚拟网络直接连到物理 LAN 的方式. "mode"指转发模式。

(1) mode='nat' : 所有连接到该虚拟网络的虚拟的网络都会经过物理机器的网卡, 并转换成物理网卡的地址。



```

<network>
  <name>default</name>
  <bridge name="virbr0" />
  <forward mode="nat" />
  <ip address="192.168.122.1"
netmask="255.255.255.0">
    <dhcp>
      <range start="192.168.122.2"
end="192.168.122.254" />
    </dhcp>
  </ip>
  <ip family="ipv6"
address="2001:db8:ca2:2::1" prefix="64" />
</network>

```



也可以指定公共的IP地址和端口号。

```

<forward mode='nat'><nat><address start='1.2.3.4'
end='1.2.3.10' /> </nat> </forward>
<forward mode='nat'><nat><port start='500'
end='1000' /></nat></forward>

```

(2) mode='route' : 类似于 NAT, 但是不使用NAT, 而是使用routing table。



```

<network>
  <name>local</name>
  <bridge name="virbr1" />
  <forward mode="route" dev="eth1" />
  <ip address="192.168.122.1"
netmask="255.255.255.0">
    <dhcp>
      <range start="192.168.122.2"
end="192.168.122.254" />
    </dhcp>
  </ip>
  <ip family="ipv6"

```

```
address="2001:db8:ca2:2::1" prefix="64" />
</network>
```

(3) mode='bridge': 使用不受libvirt管理的bridge, 比如主机上已有的bridge; open vswitch bridge; 使用 macvtap's "bridge" 模式

```
<network>
  <name>host-bridge</name>
  <forward mode="bridge"/>
  <bridge name="br0"/>
</network>
```

(4) mode='passthrough': 使用 a macvtap "direct" connection in "passthrough" mode 指定主机上的特定网卡用于虚拟网络

```
<forward mode='passthrough'>
  <interface dev='eth10' />
  <interface dev='eth11' />
  <interface dev='eth12' />
  <interface dev='eth13' />
  <interface dev='eth14' />
</forward>
```

(5) mode='hostdev': 直接分配主机上的网络设备。

```
<forward mode='hostdev' managed='yes'>
  <driver name='vfio' />
  <address type='pci' domain='0' bus='4'
slot='0' function='1' />
  <address type='pci' domain='0' bus='4'
slot='0' function='2' />
  <address type='pci' domain='0' bus='4'
slot='0' function='3' />
</forward>
```

详细的 XML 定义说明在 <https://libvirt.org/format.html>。

1.3 Libvirt API 的实现


libvirt API 的实现是在各个 Hypervisor driver 和 Storage driver 内。Hypervisor 驱动包括：

- **LXC** - Linux Containers
- **OpenVZ**
- **QEMU**
- **Test** - Used for testing
- **UML** - User Mode Linux
- **VirtualBox**
- **VMware ESX**
- **VMware Workstation/Player**
- **Xen**
- **Microsoft Hyper-V**
- **IBM PowerVM (phyp)**
- **Parallels**
- **Bhyve** - The BSD Hypervisor

1.4 Libvirt 的 Python 绑定


python-libvirt 包含 Libvirt 的 Python 语言绑定。安装 libvirt 时, 默认会安装 python-libvirt。来源: <https://libvirt.org/python.html> <https://pypi.python.org/pypi/libvirt-python>

Python API 和 C API 之间几乎是一一对应的映射关系, 比如:



```
#C API
int virConnectNumOfDomains (virConnectPtr conn);
int virDomainSetMaxMemory (virDomainPtr domain, unsigned long
memory);

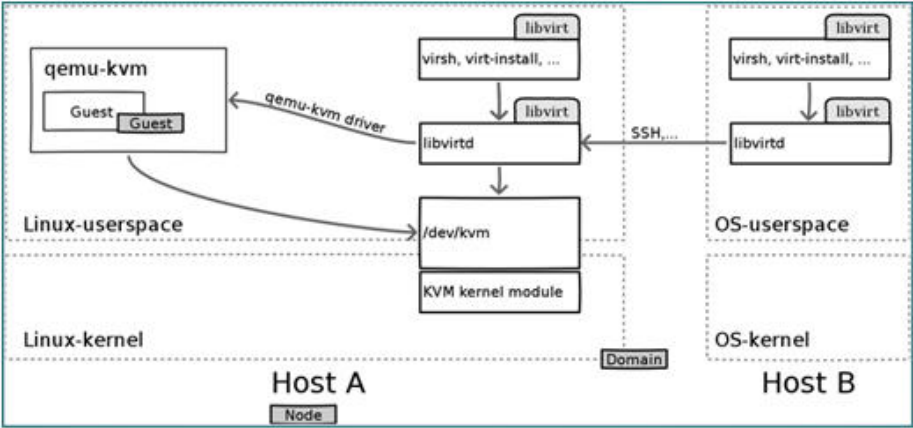
#Python API
virConnect::numOfDomains(self)
virDomain::setMaxMemory(self, memory)
```



因此，libvirt 官网并没有提供详细的 python API 描述。

2. QEMU/KVM libvirt 驱动

2.1 架构



- Libvirtd 是一个 daemon 进程，可以被本地的virsh调用，也可以被远程的virsh调用
- Libvirtd 调用 qemu-kvm 操作KVM 虚拟机

这里有一个 virsh 命令、Libvirt C API、 QEMU driver 方法 和 QEMU Monitor 命令的对照表（部分）：

virsh command	Public API	QEMU driver function	Monitor command
virsh create XMLFILE	virDomainCreateXML()	qemudDomainCreate()	info cpus, cont, change vnc password, balloon (all indirectly)
virsh suspend GUEST	virDomainSuspend()	qemudDomainSuspend()	stop
virsh resume GUEST	virDomainResume()	qemudDomainResume()	cont
virsh shutdown GUEST	virDomainShutdown()	qemudDomainShutdown()	system_powerdown
virsh setmem GUEST MEM-KB	virDomainSetMemory()	qemudDomainSetMemory()	balloon (indirectly)
virsh dominfo GUEST	virDomainGetInfo()	qemudDomainGetInfo()	info balloon (indirectly)
virsh save GUEST FILENAME	virDomainSave()	qemudDomainSave()	stop, migrate exec
virsh restore FILENAME	virDomainRestore()	qemudDomainRestore()	cont
virsh dumpxml GUEST	virDomainDumpXML()	qemudDomainDumpXML()	info balloon (indirectly)
virsh attach-device	virDomainAttachDevice()	qemudDomainAttachDevice()	change, eject, usb_add, pci_add

GUEST XMLFILE	chDevice()	achDevice()	(all indirectly)
virsh detach-device GUEST XMLFILE	virDomainDetachDevice()	qemudDomainDetachDevice()	pci_del (indirectly)
virsh migrate GUEST DEST-URI	virDomainMigrate()	qemudDomainMigratePerform()	stop, migrate_set_speed, migrate, cont
virsh domblkstat GUEST	virDomainBlockStats()	qemudDomainBlockStats()	info blockstats
-	virDomainBlockPeek()	qemudDomainMemoryPeek()	memsave

2.2 安装

有三种方式来安装 libvirt :

- (1) 下载 libvirt 的源代码 , 然后编译和安装
- (2) 从各 Linux 的发行版中直接安装 , 比如 Ubuntu 上运行 apt-get install libvirt-bin
- (3) 从 git 上克隆 libvirt 的代码 , 然后编译和安装

2.3 libvirt log

这篇文章 描述了 libvirt log。设置所有日志的方法是在 /etc/libvirt/libvirtd.conf 中添加下面的配置然后重启 libvirt :

```
log_filters="1:libvirt 1:util 1:qemu"  
log_outputs="1:file:/var/log/libvirt/libvirtd.log"
```

3 使用 libvirt 编程来管理 KVM 虚机的实例

这里只描述基本的过程。具体的过程 , 下一篇文章会具体分析 Nova 中 libvirt 的使用。

- 1. 定义虚机的基本配置 , 包括 vCPU、内存、磁盘或者cdrom以及启动顺序 , 生成 xml 配置 , 调用 virDomainCreateXML API 启动一个虚拟机
- 2. 使用 Domain 相关的 API 来管理虚拟机的生命周期。我的这篇文章有虚拟机生命周期的详细介绍。
- 3. 添加磁盘 : 定义一个 disk 的 xml 配置 , 使用 virDomainAttachDevice API 将它挂载到虚拟机上。如果不是本地的源磁盘 , 需要提前准备好。
- 4. 添加interface : 使用 Network API 定义一个虚拟网络 (需要提前准备好物理网络) , 然后定义一个 interface 的 XML 配置 , 使用 virDomainAttachDevice API 将它加到虚拟机。
- 5. 按照需要 , 重复2、3、4步骤。

分类: KVM,虚拟化

好文要顶

关注我

收藏该文

SammyLiu
关注 - 30
粉丝 - 470

荣誉 : 推荐博客

+加关注

3


0

推荐


反对

« 上一篇 : KVM 介绍 (4) : I/O 设备直接分配和 SR-IOV [KVM PCI/PCIe Pass-Through SR-IOV]
» 下一篇 : IBM 云的商业动作之我见 (1) : IBM 收购 OpenStack 托管私有云公司 Blue Box [IBM Acquired Blue Box]


评论:

#1楼 2017-02-09 09:42 | [divlee](#) 

写的太好了加油

[支持\(0\)](#) [反对\(0\)](#)#2楼[楼主] 2017-02-09 23:16 | [SammyLiu](#) @ [divlee](#)

很高兴它对你有点价值，请多提宝贵意见。

[支持\(0\)](#) [反对\(0\)](#)#3楼 2017-04-01 10:38 | [大相林](#) 

群主非常高兴看到您的博客，更期待大家创建一个渠道比如微信、QQ方便沟通，与咨询。期待大家一起学习的道路上共勉。

[支持\(0\)](#) [反对\(0\)](#)[刷新评论](#) [刷新页面](#) [返回顶部](#)注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#)网站首页。

【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库

【报表】Excel 报表开发18 招式，人人都能做报表

【活动】阿里云海外云服务全面降价助力企业全球布局

【实用】40+篇云服务器操作及运维基础知识！



最新IT新闻:

- 知乎上线视频功能，以后看教程更方便了
 - 一年只赚2万元：乐视游戏或被出售
 - Unity获得4亿美元投资，现估值为26亿美元
 - 直播对陌陌的意义，就像王者荣耀之于腾讯游戏
 - 死磕支付宝？苏宁金融发布“星辰计划”：扫码支付返888元
- » [更多新闻...](#)



最新知识库文章:

- [程序员的工作、学习与绩效](#)
- [软件开发为什么很难](#)
- [唱吧DevOps的落地，微服务CI/CD的范本技术解读](#)
- [程序员，如何从平庸走向理想？](#)
- [我为什么鼓励工程师写blog](#)

» [更多知识库文章...](#)

Powered by: [博客园](#) 模板提供 : [沪江博客](#) Copyright ©2017 SammyLiu