# **第二周：****云计算及虚拟化**

### **注意：从本节开始，将不会直接提供完整的实现代码，主要提供设计思路及范例代码以引导你独立完成模块的实现。如果在代码实现中遇到问题，请在课程问答里提出。**

## **一、实验简介**

本节实验为实验楼训练营课程《仿OpenStack开发云计算管理软件》第2节，希望与大家一起完成下面几项任务：

1. 云计算及虚拟化基本概念；
2. 虚拟化技术学习；
3. 虚拟化开发 Libvirt API ；
4. 虚拟机管理脚本开发；

## **二、什么是云计算？**

云计算的概念请阅读资料：<http://baike.baidu.com/view/1316082.htm>。实验文档中不做展开介绍，任何疑问请随时在实验文档下方的课程问答中提出。

本项目实现的云计算管理平台属于IaaS（基础设施即服务）软件，IaaS的概念是将物理资源进行整合，通过网络的方式提供虚拟资源。在我们的项目中，物理资源指的是服务器，存储，网络等，虚拟资源指的是整合了虚拟CPU，虚拟内存，虚拟磁盘及虚拟网络的虚拟机。用户可以按需的随时创建和释放虚拟机，而不需要关心虚拟机具体运行在哪一台服务器以及存储在哪一块磁盘上，这就是云计算的服务模式。

## **三、虚拟化技术**

### **3.1 虚拟化概述**

虚拟化技术的入门请阅读资料：[虚拟化技术漫谈](http://www.ibm.com/developerworks/cn/linux/l-cn-vt/index.html)。任何疑问请随时在实验文档下方的课程问答中提出。

企业领域采用的虚拟化技术大多是服务器虚拟化，大多需要CPU上硬件支持虚拟化技术（例如Intel VT）的支持，Intel VT等技术在硬件层面支持了虚拟化的实现，提升了虚拟机的性能，同时可以运行未经修改过的原生内核（例如Window虚拟机）。

目前主流的商业虚拟化产品是VMware ESXi，微软的Hyper-V，开源领域最常见的是Xen及KVM。国内的公有云，例如阿里云，青云等都采用Xen或KVM作为虚拟化平台。

本实验采用的虚拟化技术是QEMU，实现的接口同样适用于非常流行的KVM虚拟化技术。KVM和QEMU是相辅相成的，QEMU可以使用KVM内核模块加速，而KVM需要使用QEMU运行虚拟机。KVM是在内核态的，QEMU是在用户态。

虚拟化技术并不复杂，Xen/KVM都是由Linux内核中的模块提供支持，感兴趣的同学可以了解下：[探索Linux内核虚拟机](http://www.ibm.com/developerworks/cn/linux/l-linux-kvm/index.html).

### **3.2 虚拟化管理**

本实验中实现的虚拟化管理模块基于Libvirt API（将在下一节详细介绍）。

虚拟化管理包含下述两部分的管理：

1. 物理资源：计算资源（服务器），存储资源，网络资源
2. 虚拟资源：虚拟机（含虚拟CPU，虚拟磁盘，虚拟内存等），镜像，模板，虚拟网络

#### **3.2.1 物理资源管理**

服务器管理比较简单，云是一个资源池，只需要服务器加入云的资源池或者从云资源池中移除两种操作。

云所使用的物理存储有多种架构模式，最常见的是分布式文件系统，通过多台存储服务器构建成分布式文件系统，用来存储虚拟机的虚拟磁盘，提供给运行虚拟机计算服务器使用，大部分分布式文件系统会提供数据冗余功能，以保证单个存储节点挂掉时数据的可用性。目前比较流行的分布式存储是ceph，moosefs等，感兴趣可以查看ceph的介绍（[Ceph：一个 Linux PB 级分布式文件系统](http://www.ibm.com/developerworks/cn/linux/l-ceph/)）。通过把存储系统挂载到每个计算服务器上，并添加到云平台管理中进行统一管理。

网络资源则很复杂，物理的网络环境在机房中会分层设置，云平台则需要对物理网络到虚拟网络的映射关系进行管理。

在LouCloud中，为了简化实现，只提供添加和移除服务器的功能，存储则直接使用服务器的本地硬盘，这不需要搭建分布式文件系统，但需要将虚拟机的镜像拷贝到每个服务器上，并放置到同样的目录下。物理网络我们统一在每个服务器上配置虚拟机使用NAT方式，即每个服务器内部创建一个虚拟交换机，虚拟机使用内部IP地址，连接虚拟交换机通过NAT的方式连接外部网络。

添加服务器的过程中需要对服务器上的Libvirt服务进行测试，毕竟我们最终会通过Libvirt API来进行虚拟机的管理。

删除服务器时需要保证服务器上没有任何虚拟机运行，否则会造成虚拟机状态的异常。

### **3.2.2 虚拟资源管理**

虚拟机需要运行在服务器上，这是基本原理。虚拟资源的管理包含镜像管理，模板管理及虚拟机管理。

镜像用来创建虚拟机的磁盘，模板是一个虚拟机的基本配置，包含磁盘镜像及CPU个数，内存大小。

镜像本身是一个虚拟磁盘，虚拟磁盘有很多种格式，比如RAW，ISO，QCOW2等，LouCloud选择使用QCOW2格式的虚拟磁盘，因为QCOW2格式可以通过快照方式创建新的磁盘，去掉了虚拟机创建过程中复制磁盘文件的时间。QCOW2和RAW格式的对比可以参考文章：[QEMU 使用的镜像文件：qcow2 与 raw](http://www.ibm.com/developerworks/cn/linux/1409_qiaoly_qemuimgages/)。

LouCloud需要在多台服务器中选择一台运行用户请求创建的虚拟机，这个选择的算法可以简单地实现为哪一台服务器上运行的虚拟机数最少就选择哪一台。当然，实际的云环境中为了达到资源的最优配比这个选择算法可能会很复杂。

虚拟机创建过程中只需要选择模板，并输入虚拟机的名称即可，LouCloud将模板转换为Libvirt API的参数，连接到选择的服务器上进行虚拟机的创建。我们已经选择采用服务器的本地磁盘，所以镜像应该放置到每一个服务器的固定目录中，虚拟机才可以成功创建并启动。

虚拟机的其他操作，例如开关机，重启，删除，都比创建简单很多，只需要调用Libvirt相应的API就可以达到目的。下面我们开始介绍Libvirt API。

## **四、Libvirt API**

### **4.1 Libvirt简介**

Libvirt是一套免费、开源的支持Linux下主流虚拟化工具的C函数库，其旨在为包括Xen在内的各种虚拟化工具提供一套方便、可靠的编程接口，支持与C,C++,Ruby,Python等多种主流开发语言的绑定。当前主流Linux平台上默认的虚拟化管理工具virt-manager(图形化),virt-install（命令行模式）等均基于libvirt开发而成。

Libvirt 库是一种实现 Linux 虚拟化功能的 Linux API，它支持各种虚拟机监控程序，包括 Xen 和 KVM，以及 QEMU 和用于其他操作系统的一些虚拟产品。

简单的说Libvirt是一套标准化虚拟化管理接口，可以管理上述我们提到的各种虚拟资源。Libvirt除了各种语言的SDK外，还提供一个命令行工具virsh。在我们的实验环境中可以通过下列方式来启动，请花些时间熟悉virsh命令，非常有助于我们后续开发过程中的调试。

# 启动libvirt-bin服务

sudo service libvirt-bin start

# 查看当前系统中的虚拟机列表，默认返回为空

sudo virsh list

# 查看当前系统中虚拟网络列表，默认返回default网络

sudo virsh net-list --all

virsh常用命令列表：

命令 说明

help 显示该命令的说明

quit 结束 virsh，回到 Shell

connect 连接到指定的虚拟机服务器

create 启动一个新的虚拟机

destroy 删除一个虚拟机

start 开启（已定义的）非启动的虚拟机

define 从 XML 定义一个虚拟机

undefine 取消定义的虚拟机

dumpxml 转储虚拟机的设置值

list 列出虚拟机

reboot 重新启动虚拟机

save 存储虚拟机的状态

restore 回复虚拟机的状态

suspend 暂停虚拟机的执行

resume 继续执行该虚拟机

dump 将虚拟机的内核转储到指定的文件，以便进行分析与排错

shutdown 关闭虚拟机

setmem 修改内存的大小

setmaxmem 设置内存的最大值

setvcpus 修改虚拟处理器的数量

### **4.2 Libvirt配置**

Libvirt两个主要的配置文件是/etc/libvirt/libvirt.conf和/etc/libvirt/qemu.conf

Libvirt默认运行在root，用户组libvirtd权限下，所以以shiyanlou用户运行virsh list会报错。可以通过更新/etc/libvirt/qemu.conf里的user和group修改。在配置文件下还有很多其他的选项，可以阅读配置文件的注释进行学习。

**这里需要记住，一旦修改配置文件，一定要重启libvirt-bin：**

sudo service libvirt-bin stop

sudo service libvirt-bin start

为了方便以后的操作及开发，我们将shiyanlou用户添加到libvirtd用户组中，使shiyanlou用户也可以运行virsh命令。

添加方式为以下命令:

sudo adduser shiyanlou libvirtd

**注意:**添加后需要重新连接字符模式（刷新）或者在图形模式下执行ssh localhost再创建新的session才生效，是否生效可以使用groups命令查看，如果结果中没有libvirtd则说明当前session中还没与生效。

### **4.3 Libvirt Domain XML**

Libvirt可以管理多种组件：虚拟机，网络，网络过滤器，存储，快照等等。每种组件都通过XML文件进行定义和描述，组件通常有一系列相应的配置和功能，我们的LouCloud只会用到其中最常见的一小部分，大部分内容可以查看 Libvirt 文档学习。

本节简要介绍 Libvirt 虚拟机的配置文件，在 Libvirt 中虚拟机成为 Domain 。 Domain 的 XML 配置选项详细文档见：<http://libvirt.org/formatdomain.html>。

Domain XML配置实例以及注释说明（该配制信息后面会用到）：

<!-- 虚拟化的类型，可以使qemu，kvm等 -->

<domain type='qemu'>

<!-- 虚拟机名称 -->

<name>test</name>

<!-- 虚拟CPU个数 -->

<vcpu>1</vcpu>

<!-- 虚拟内存数，默认为KB -->

<memory>102400</memory>

<!-- 操作系统类型 -->

<os>

<type arch='x86\_64' machine='pc-1.0'>hvm</type>

<boot dev='hd'/>

</os>

<!-- 虚拟硬件的功能表 -->

<features>

<acpi/>

<apic/>

<pae/>

</features>

<!-- 虚拟硬件上的时钟配置 -->

<clock offset='utc'/>

<!-- 虚拟机内部关机时自动释放资源 -->

<on\_poweroff>destroy</on\_poweroff>

<on\_reboot>restart</on\_reboot>

<!-- 系统崩溃时自动重启 -->

<on\_crash>restart</on\_crash>

<!-- 虚拟设备列表-->

<devices>

<!-- 虚拟设备模拟器路径 -->

<emulator>/usr/bin/qemu-system-x86\_64</emulator>

<!-- 虚拟磁盘配置 -->

<disk type='file' device='disk'>

<!-- 虚拟磁盘的驱动类型和格式-->

<driver name='qemu' type='qcow2'/>

<!-- 虚拟磁盘的路径 -->

<source file='/tmp/test.qcow2'/>

<!-- 虚拟磁盘连接方式-->

<target dev='hda' bus='ide'/>

<address type='drive' controller='0' bus='0' target='0' unit='0'/>

</disk>

<!-- 虚拟网络配置 -->

<interface type='network'>

<!-- 虚拟网络名称 -->

<source network='default'/>

<!-- 虚拟网络驱动方式 -->

<model type='virtio'/>

<!-- 虚拟网络过滤器配置 -->

<filterref filter='clean-traffic'/>

</interface>

<!-- 虚拟机界面连接方式选用VNC，VNC端口设置为随机产生 -->

<graphics type='vnc' port='-1' autoport='yes'/>

<!-- 输入配置 -->

<input type='tablet' bus='usb'/>

<memballoon model='virtio'>

<address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>

</memballoon>

</devices>

</domain>

这个配置文件可以用来创建一个名字为test的QEMU虚拟机，同时具备1个虚拟CPU和100M内存，并提供VNC的连接方式。虚拟机的磁盘采用QCOW2格式的磁盘，VNC的端口号需要在虚拟机创建成功后随机分配。

### **4.4 服务器连接**

virsh连接服务器需要认证，最常见的认证方式为TLS和SSH，详细介绍见：<http://libvirt.org/remote.html>。

我们选择简单的SSH认证方式，每个服务器都需要提供一个URI的字符串才可以用来连接，字符串中制定了虚拟化类型，连接认证方式和连接的服务器地址。

我们尝试使用SSH方式连接本机，本机采用QEMU虚拟化，因此根据libvirt的URI定义本机虚拟化链接为：qemu+ssh://shiyanlou@localhost/system

# 本地执行list命令，连接前确保启动libvirt-bin服务

virsh list

# 远程连接执行list命令，需要输入shiyanlou用户的密码:shiyanlou

virsh -c qemu+ssh://shiyanlou@localhost/system list

在Libvirt Python API中进行连接，示例代码如下：

shiyanlou:~/ $ python

Python 2.7.6 (default, Jun 22 2015, 17:58:13)

[GCC 4.8.2] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>> import libvirt

>>> hostUri = 'qemu+ssh://shiyanlou@localhost/system'

>>>

>>> host = libvirt.open(hostUri)

shiyanlou@localhost's password: #此处需要输入shiyanlou密码

>>> host

<libvirt.virConnect object at 0x7ffff7e44a50>

>>> dir(host) # 查看host的接口

上面得到的host即为libvirt的一个服务器连接对象，可以用来创建和管理虚拟机。而LouCloud的服务器管理就是管理若干个host对象。host的接口非常丰富，不只有各种创建虚拟机的方式，也有对其他资源的配置和管理API。请查阅libvirt.org [API 文档](http://libvirt.org/html/index.html) 进行详细学习，文档内容很多，建议大家可以学习本实验的过程中多多查阅，尤其是libvirt-domain部分。API 文档为C语言接口文档，但与Python映射关系非常简单，关联方式见<http://libvirt.org/python.html>。

### **4.5 虚拟机操作**

现在我们开始使用Libvirt创建和管理虚拟机了。

虚拟机的创建是最复杂的一个过程，我们由于系统还没有实现模板，镜像等操作，所以创建的每个步骤都将由我们手动操作，也更容易接触到细节。

### **4.5.1 创建**

在4.4中我们有了一个 Libvirt 的 host对象，即虚拟机创建的服务器已经有了，除此之外，我们还需要下面的内容：

1. 虚拟机的配置
2. 虚拟机的磁盘
3. 虚拟机的网络
4. 虚拟机的连接方式

首先虚拟机的配置：名称louvm1，1个虚拟CPU，100M虚拟内存。

其次，虚拟磁盘我们选用QCOW2格式，qemu有一个自带的命令qemu-img用来管理虚拟磁盘，可以通过man qemu-img查看基本用法，此处我们用qemu-img为虚拟机创建一个磁盘，为了能够启动虚拟机，我们选择使用现成的镜像，而不是一个空磁盘（需要ISO文件来安装系统）。为了节约时间，我们选用比较小的操作系统镜像-FreeDOS镜像，将在后续的实验中多次用到。

cd /home/shiyanlou

# 下载镜像

wget http://labfile.oss.aliyuncs.com/courses/354/FreeDOS.qcow2.bz2

# 解压镜像

bzip2 -d FreeDOS.qcow2.bz2

# 使用qemu-img查看FreeDOS.qcow2镜像信息，展示虚拟磁盘的实际大小和最大空间，格式等信息

qemu-img info FreeDOS.qcow2

# 使用qemu-img由镜像FreeDOS.qcow2创建快照作为虚拟机的虚拟磁盘louvm1.qcow2

qemu-img create -f qcow2 louvm1.qcow2 -b FreeDOS.qcow2

# 查看虚拟磁盘的配置

qemu-img info louvm1.qcow2

现在我们已经有了虚拟机的磁盘，下一步虚拟机的网络我们选择default默认网络，这个网络的信息可以通过virsh net-dumpxml default查看发现是一个NAT的简单网络。不过由于实验楼环境限制，default网络无法激活，在此我们暂时去掉了虚拟机的网络配置。

虚拟机的桌面连接方式我们选用VNC，即使虚拟机没有IP地址也可以连接，因为这个VNC是QEMU提供的内置模式，虚拟机只要处于运行状态就可以连接。

现在我们有了所有的轮子，可以组装成一个虚拟机的配置文件louvm1.xml：

cd /home/shiyanlou/Code/shiyanlou\_cs354

mkdir virtscripts

cd virtscripts

vim louvm1.xml

输入下面的配置信息，配置信息中前面已经有了介绍，如果不清楚可以查看libvirt详细文档，或者到课程问答中提问。

<domain type='qemu'>

<name>louvm1</name>

<vcpu>1</vcpu>

<memory>102400</memory>

<os>

<type arch='x86\_64' machine='pc-1.0'>hvm</type>

<boot dev='hd'/>

</os>

<features>

<acpi/>

<apic/>

<pae/>

</features>

<clock offset='utc'/>

<on\_poweroff>destroy</on\_poweroff>

<on\_reboot>restart</on\_reboot>

<on\_crash>restart</on\_crash>

<devices>

<emulator>/usr/bin/qemu-system-x86\_64</emulator>

<disk type='file' device='disk'>

<driver name='qemu' type='qcow2'/>

<source file='/home/shiyanlou/louvm1.qcow2'/>

<target dev='hda' bus='ide'/>

<address type='drive' controller='0' bus='0' target='0' unit='0'/>

</disk>

<graphics type='vnc' port='-1' autoport='yes'/>

<input type='tablet' bus='usb'/>

<memballoon model='virtio'>

<address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>

</memballoon>

</devices>

</domain>

我们首先通过virsh命令创建虚拟机：

# 查看所有虚拟机

virsh list --all

# 由配置文件louvm1.xml创建虚拟机

virsh define louvm1.xml

# 查看所有虚拟机,louvm1为关机状态

virsh list --all

# 启动louvm1虚拟机

virsh start louvm1

# 查看所有虚拟机，louvm1为运行状态

virsh list --all

使用Libvirt API进行创建可以使用如下代码，由于虚拟机无法重名或共用一个镜像，所以需要将上述虚拟机先删除才可以，删除方法见4.5.6删除一节。

下面的API调用需要用到4.4的方法获取到的host。

# 首先读取 /home/shiyanlou/Code/shiyanlou\_cs354/virtscripts/louvm1.xml 文件内容到字符串 domainXMLString

# 然后调用下面的 API 在服务器 host 上创建虚拟机，第二个参数为创建后的行为标志，0为默认行为，即创建后启动虚拟机，其他参数见[详细介绍](http://libvirt.org/html/libvirt-libvirt-domain.html#virDomainCreateFlags)，createXML 将创建并运行虚拟机 louvm1，如果只创建不运行可以选择另外一个接口 `defineXML()`

>> domain = host.createXML(domainXMLString, 0)

# 查看虚拟机状态，state()函数的参数及返回值请参考文档

>> domain.state()

# 查看虚拟机是否已启动

>> domain.state()[0] == libvirt.VIR\_DOMAIN\_RUNNING

# 查看domain的接口

>> dir(domain)

上面得到的domain即为libvirt的一个虚拟机对象，可以用来执行虚拟机的各种操作。而LouCloud的虚拟机管理就是管理若干个domain对象。domain的接口非常丰富。请查阅libvirt.org [API 文档](http://libvirt.org/html/index.html)进行详细学习。

请自己独立完成虚拟机创建操作的函数createVM()，并放置到目录/home/shiyanlou/Code/shiyanlou\_cs354/virtscripts/vm\_utils.py文件中，函数参数：

1. 虚拟机名称
2. 虚拟机CPU数量
3. 虚拟机内存数
4. 虚拟机磁盘位置

函数返回为True/False表示虚拟机是否创建成功。

### **4.5.2 关机**

关机操作命令为virsh shutdown，强制关机操作的virsh命令如下，后者相当于拔电源：

这里需要注意，如果是直接由createXML()创建的虚拟机，则在关机时就自动从Libvirt系统中清除了。只有经过defineXML()函数创建的虚拟机关机后才可以继续start()启动，否则还是需要重新执行createXML()来再次创建并启动虚拟机。

virsh destroy louvm1

关机操作API范例如下：

# domain 为 Libvirt 的虚拟机对象

# 根据名字查找虚拟机

>> domain = host.lookupByName(vmname)

# 强制关机

>> domain.destroy()

# 查看状态，会发现第一个数字变为了5，即关机状态

>> domain.state()

请自己独立完成虚拟机关机操作的函数shutdownVM()，并放置到目录/home/shiyanlou/Code/shiyanlou\_cs354/virtscripts/vm\_utils.py文件中，函数参数：

1. 虚拟机名称

函数返回为True/False表示虚拟机是否关闭成功，采用强制关机方案。

### **4.5.3 开机**

开机操作命令为virsh start：

virsh start louvm1

开机操作API范例如下：

# domain 为 Libvirt 的虚拟机对象

>> domain.create()

# 查看虚拟机状态，state()函数的参数及返回值请参考文档

>> domain.state()

请自己独立完成虚拟机启动操作的函数startVM()，并放置到目录/home/shiyanlou/Code/shiyanlou\_cs354/virtscripts/vm\_utils.py文件中，函数参数：

1. 虚拟机名称

函数返回为True/False表示虚拟机是否启动成功。

### **4.5.4 连接**

虚拟机使用VNC进行连接，当虚拟机启动成功后，可以使用下述命令查询虚拟机的信息：

virsh dumpxml louvm1

在返回的XML文件中会发现，先前的VNC端口-1已经被替换为一个正数，这就是随机分配的VNC端口，我们可以使用vncviewer连接到虚拟机的桌面：

sudo apt-get install vncviewer

# VNCPORT替换为dumpxml返回的端口号

vncviewer localhost:VNCPORT

在后续的实验中，我们会选择集成NOVNC来实现在Web浏览器里连接这个桌面，感兴趣的同学可以提前看下[NoVNC](https://github.com/kanaka/noVNC)。

获取虚拟机XML字符串的函数是domain.XMLDesc()，会返回虚拟机的XML文件，从XML文件中可以提取VNC端口号，提取的方法可以使用python的XML相关的库，例如lxml的etree等。

请自己独立完成获取虚拟机VNC端口的函数getVNCPort()，并放置到目录/home/shiyanlou/Code/shiyanlou\_cs354/virtscripts/vm\_utils.py文件中，函数参数：

1. 虚拟机名称

函数返回虚拟机的VNC端口。

### **4.5.5 重启**

开机操作命令为virsh reboot，与shutdown操作相同，部分操作系统虚拟机并不支持这种软重启：

virsh reboot louvm1

重启操作对于一些操作系统虚拟机并不起作用，原因是虚拟机里没有处理Libvirt重启的信号，本实验环境中的FreeDOS虚拟机也无法由该命令进行重启。

重启操作API范例如下：

# domain 为 Libvirt 的虚拟机对象

>> domain.reboot()

# 查看虚拟机状态，state()函数的参数及返回值请参考文档

>> domain.state(0)

请自己独立完成虚拟机重启操作的函数rebootVM()，并放置到目录/home/shiyanlou/Code/shiyanlou\_cs354/virtscripts/vm\_utils.py文件中，函数参数：

1. 虚拟机名称

函数返回为True/False表示虚拟机是否启动成功。

### **4.5.6 删除**

如果虚拟机时通过createXML()创建，只需要执行domain.destroy()即为删除，如果是defineXML()创建，则需要执行domain.undefine()。删除操作命令为virsh undefine，：

# 关机

virsh destroy louvm1

# 删除虚拟机

virsh undefine louvm1

删除操作API范例如下：

# domain 为 Libvirt 的虚拟机对象

>> domain.destroy()

>> domain.undefine()

# 查看虚拟机状态，这个时候会报错，因为domain对象已失效

>> domain.state()

请自己独立完成虚拟机关机操作的函数deleteVM()，并放置到目录/home/shiyanlou/Code/shiyanlou\_cs354/virtscripts/vm\_utils.py文件中，函数参数：

1. 虚拟机名称

函数返回为True/False表示虚拟机是否删除成功。

## **五、其他资源管理**

本节实验着重介绍虚拟机的管理开发，服务器，镜像，模板都是属于系统资源，这类资源管理实现很简单，主要是数据库设计与Blueprint的实现。将在下周的实验中详细介绍。这里先放置一个引子，请大家在完成本节的虚拟化开发后思考下如何构建服务器，镜像，模板的数据库表。

## **六、本节总结及下节预告**

本节实验学习了虚拟化技术，设计并实现了服务器及虚拟机管理模块。下一节将详细介绍数据库的设计和接口开发，将本节的虚拟机存入数据库，并完成服务器，虚拟机，镜像，模板等组件的数据库表设计。

## **七、实验报告**

请按照本节实验要求完成vm\_utils.py模块，具备虚拟机的创建，开关机，重启，删除，获取VNC端口函数。步骤与设计提交LouCloud v0.2 版本到我的代码库。在实验报告中描述你的操作过程，详细描述遇到的问题及解决方案。完成后点击公开实验报告将会收到教师的点评指导。完整的参考代码本节并没有提供，如果遇到任何问题可在充分思考和搜索后，多多提问。