Python 期末作业报告

题目：云计算基础

组名： ygwSoft ¶

学号：2016329621047 姓名: 吴俚达

学号：2016329621039 姓名: 葛华盛

学号：2016339910033 姓名: 叶凯凯

题目

- 虚拟化是云计算的基础，虚拟化基础之上的自动化，才能够使这个基础真正实际运行起来。

- 现在请你用python来实现这个自动化，利用PyVIx和PySphere来操作ESXi虚拟机，实现复制、克隆、开机等一系列管理的自动化。

- 既然要能够实现管理，当然需要一个管理界面，这个UI如何实现，你可以用你拿手的任何语言。

- 希望你能够成为云计算的开发人员。

技术思路和解决方案

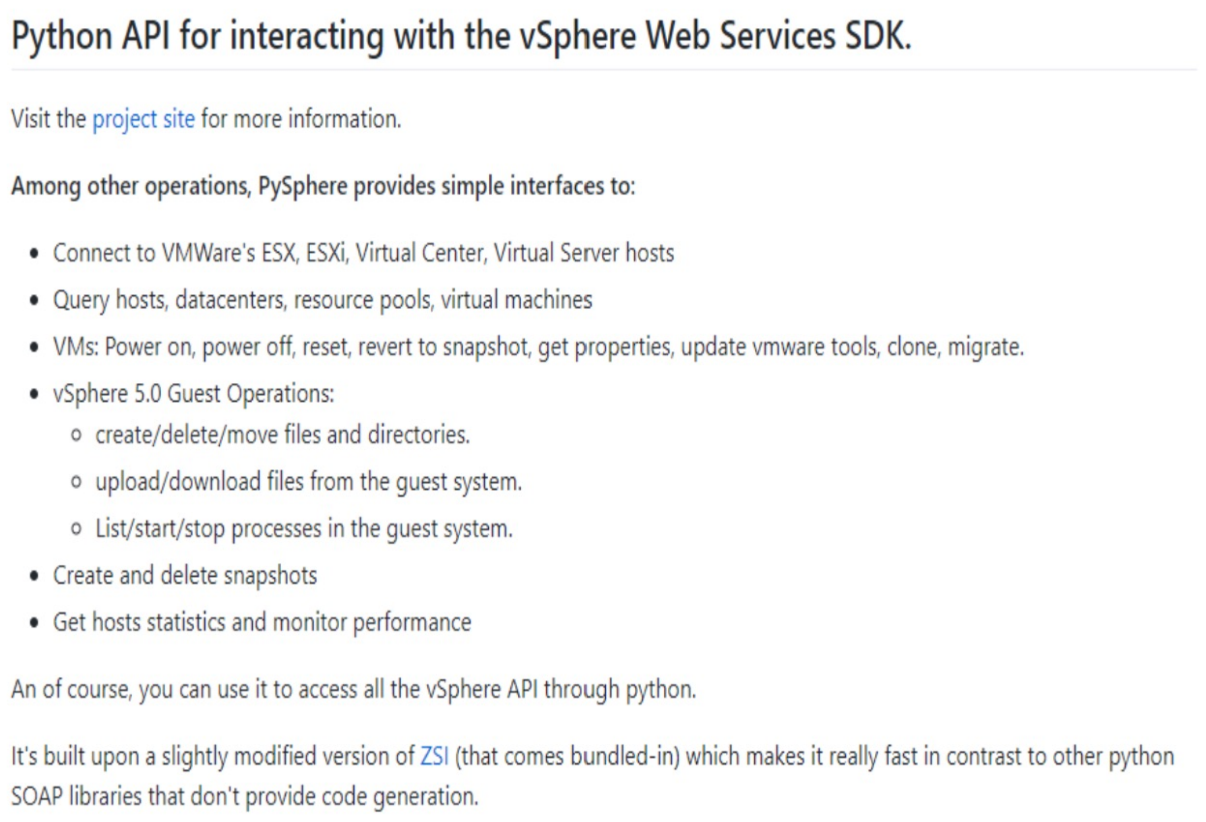
实验环境

IDE pycharm
python 2.7
pySphere 0.1.17

虚拟机工具和系统

VMware Fusion
VMware ESXi 6.5

PySphere



Python API for interacting with the vSphere Web Services SDK.

Visit the project site for more information.

Among other operations, PySphere provides simple interfaces to:

- Connect to VMWare's ESX, ESXi, Virtual Center, Virtual Server hosts
- Query hosts, datacenters, resource pools, virtual machines
- VMs: Power on, power off, reset, revert to snapshot, get properties, update vmware tools, clone, migrate.
- vSphere 5.0 Guest Operations:
  - create/delete/move files and directories.
  - upload/download files from the guest system.
  - List/start/stop processes in the guest system.
- Create and delete snapshots
- Get hosts statistics and monitor performance

An of course, you can use it to access all the vSphere API through python.

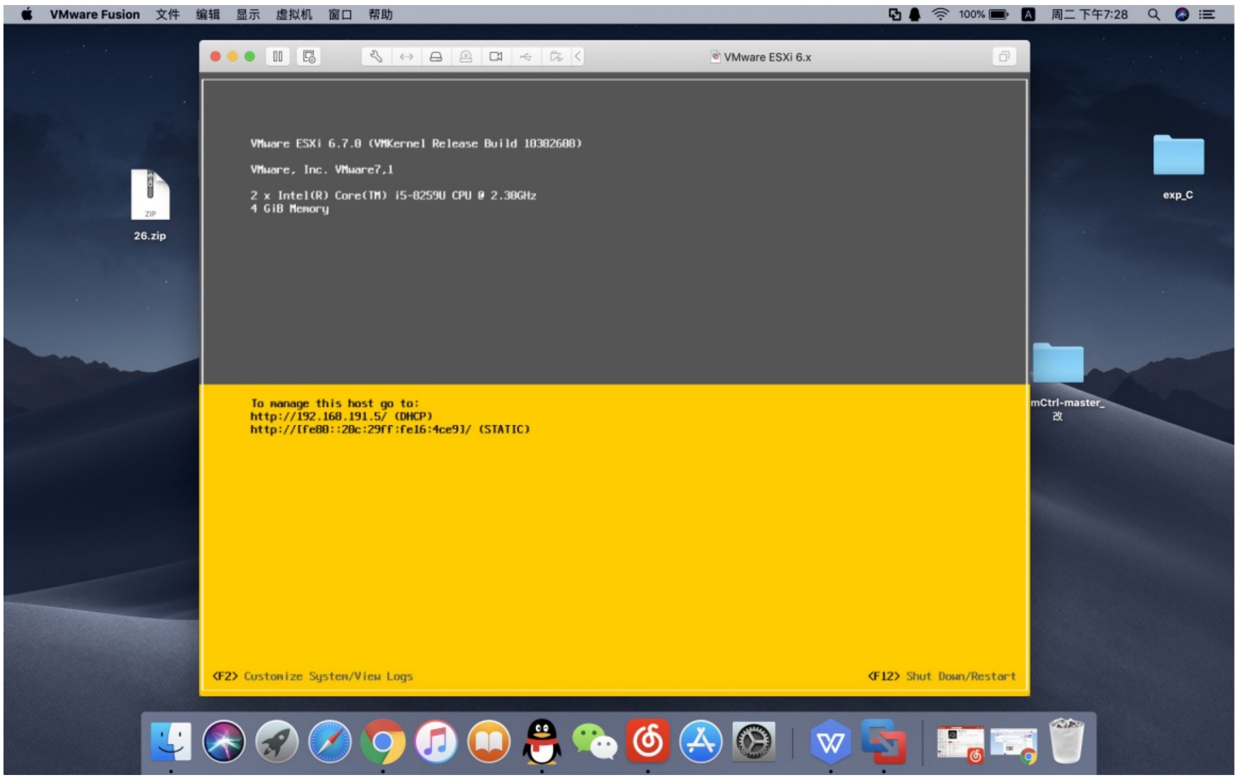
It's built upon a slightly modified version of ZSI (that comes bundled-in) which makes it really fast in contrast to other python SOAP libraries that don't provide code generation.

argparse

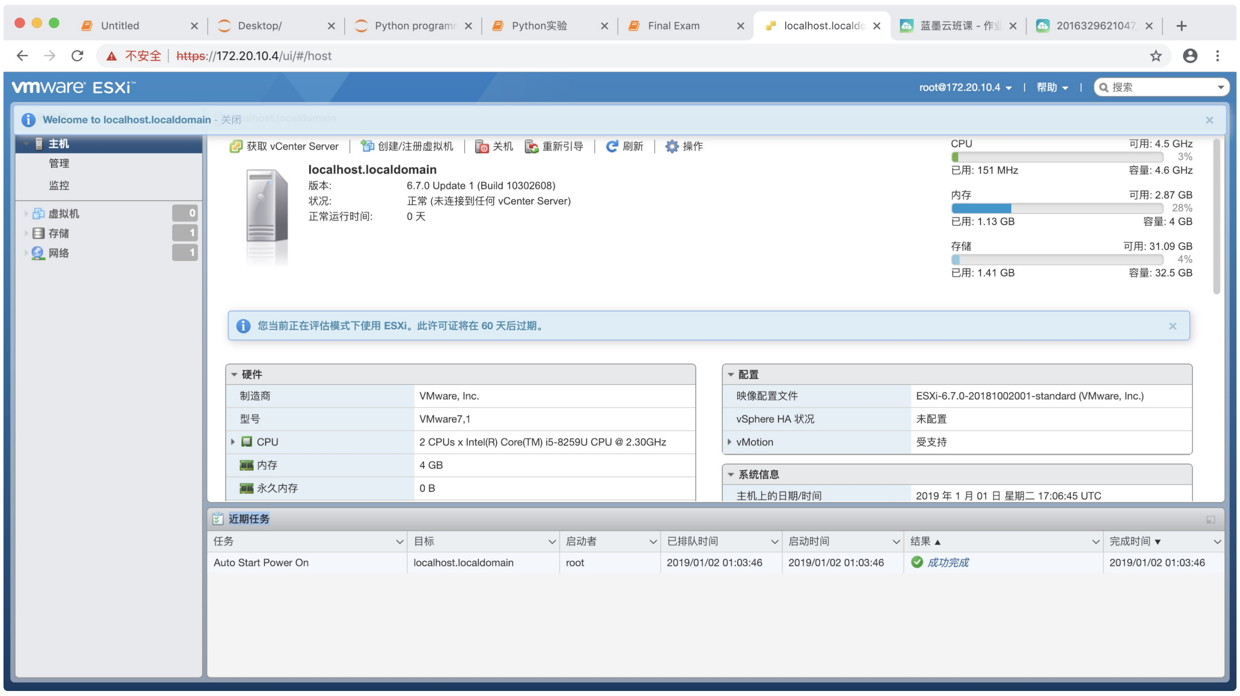
python标准库模块argparse用于解析命令行参数，编写用户友好的命令行界面，该模块还会自动生成帮助信息，并在所给参数无效时报错。 举一个小例子，在Linux系统中，我们常用到 ls 这个命令，默认情况下 ls 会显示当前目录的所有文件或文件夹名称，但是当我们想要获得更多信息，如文件的权限、建立时间等，就需要在 ls 命令后加上 -all 或其他被允许的指令，因此对 ls 这个命令而言，假设我们想改变程序的行为，显示每个文件的更多信息，而不是只显示文件名。 在这种情况下，-all 被称为可选参数。同样的， ls 还具有 -h 这个可以选择的参数，这表示打开帮助文本。 这非常有用，你可以遇到一个你以前从未使用过的程序，并且可以简单地通过阅读帮助文本来弄清楚它是如何工作的。

服务器端

在本机上的VMware Fusion安装VMware ESXi 6.5
服务端配置信息如图



```
VMware ESXi 6.7.0 (VMKernel Release Build 10302608)

VMware, Inc. VMware7,1

2 x Intel(R) Core(TM) i5-8259U CPU @ 2.30GHz
4 GiB Memory

To manage this host go to:
http://192.168.191.5/ (DHCP)
http://[fe80::20c:29ff:fe16:4ce9]/ (STATIC)

<F2> Customize System/View Logs                    <F12> Shut Down/Restart
```



Welcome to localhost.localdomain - 关闭

localhost.localdomain

版本:	6.7.0 Update 1 (Build 10302608)
状况:	正常 (未连接到任何 vCenter Server)
正常运行时间:	0 天

CPU 可用: 4.5 GHz 3% 已用: 151 MHz 容量: 4.6 GHz

内存 可用: 2.87 GB 28% 已用: 1.13 GB 容量: 4 GB

存储 可用: 31.09 GB 4% 已用: 1.41 GB 容量: 32.5 GB

您当前正在评估模式下使用 ESXi。此许可证将在 60 天后过期。

硬件	
制造商	VMware, Inc.
型号	VMware7,1
CPU	2 CPUs x Intel(R) Core(TM) i5-8259U CPU @ 2.30GHz
内存	4 GB
永久内存	0 B

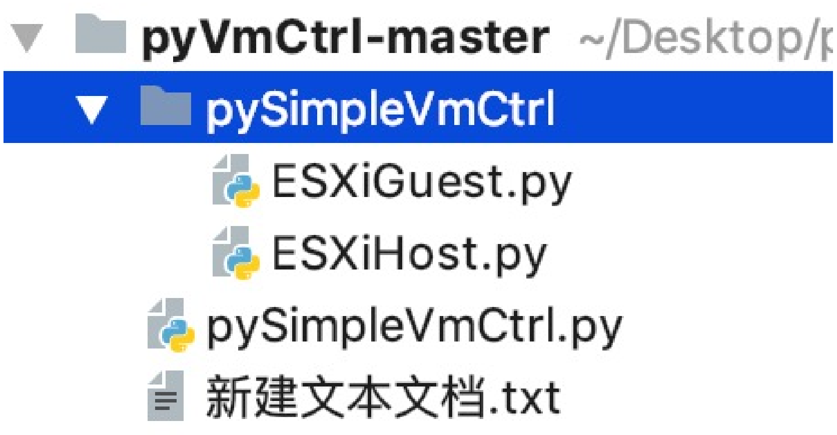
配置	
映像配置文件	ESXi-6.7.0-20181002001-standard (VMware, Inc.)
vSphere HA 状况	未配置
vMotion	受支持

系统信息	
主机上的日期/时间	2019 年 1 月 01 日 星期二 17:06:45 UTC

近期任务

任务	目标	启动者	已排队时间	启动时间	结果 ▲	完成时间 ▼
Auto Start Power On	localhost.localdomain	root	2019/01/02 01:03:46	2019/01/02 01:03:46	✓ 成功完成	2019/01/02 01:03:46

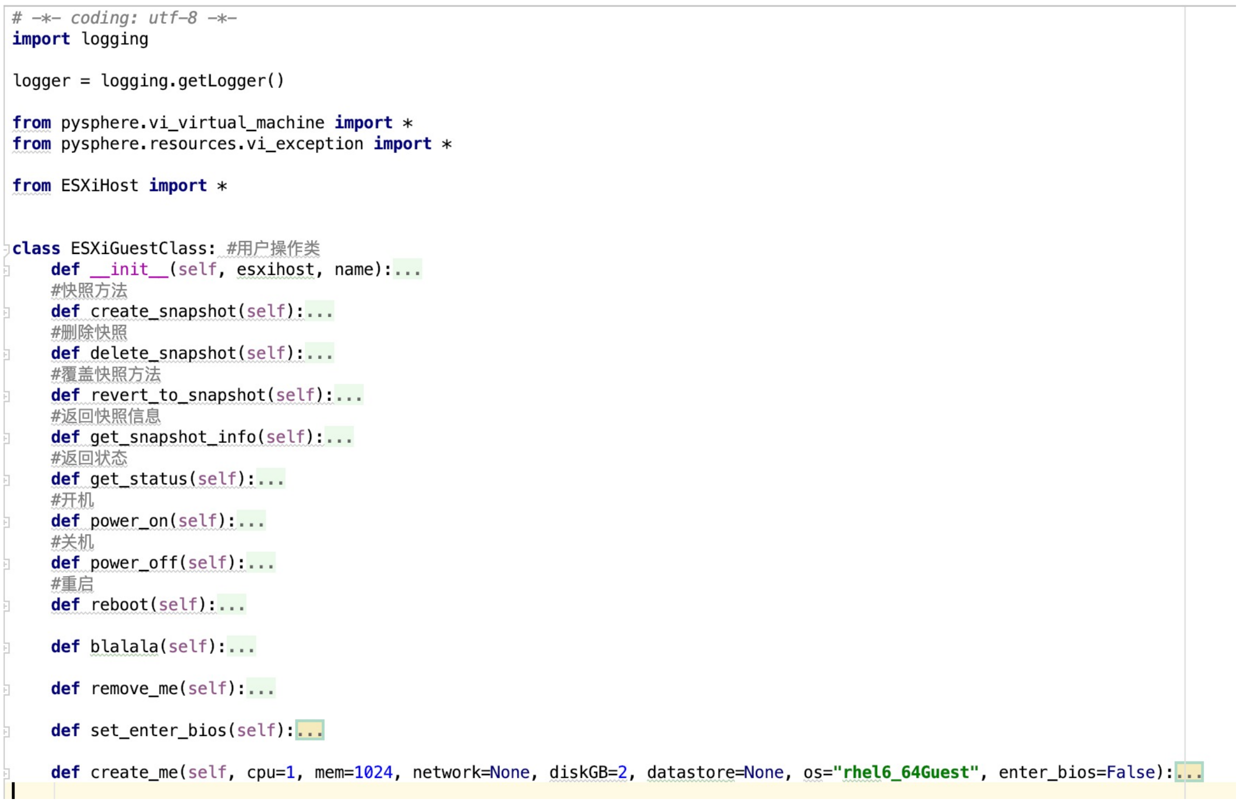
程序结构



```
▼ pyVmCtrl-master ~/Desktop/p
    ▼ pySimpleVmCtrl
          ESXiGuest.py
          ESXiHost.py
      pySimpleVmCtrl.py
      新建文本文档.txt
```

ESXiGuest.py文件写的是ESXiGuestClass用户类，针对用户对虚拟机的操作，如有快闪，开机，关机，重启等方法
ESXiHost.py文件写的是ESXiHostClass服务端类，有登陆连接服务器，配置信息查看等方法
pySimpleVmCtrl.py文件写的是命令行格式，命令行会调用用户类和服务端类的方法，用于终端命令行输入来访问，操作服务器。

详细代码在报告的附录



```python
# -*- coding: utf-8 -*-
import logging

logger = logging.getLogger()

from pysphere.vi_virtual_machine import *
from pysphere.resources.vi_exception import *

from ESXiHost import *


class ESXiGuestClass: #用户操作类
    def __init__(self, esxihost, name):...
    #快照方法
    def create_snapshot(self):...
    #删除快照
    def delete_snapshot(self):...
    #覆盖快照方法
    def revert_to_snapshot(self):...
    #返回快照信息
    def get_snapshot_info(self):...
    #返回状态
    def get_status(self):...
    #开机
    def power_on(self):...
    #关机
    def power_off(self):...
    #重启
    def reboot(self):...

    def blalala(self):...

    def remove_me(self):...

    def set_enter_bios(self):...

    def create_me(self, cpu=1, mem=1024, network=None, diskGB=2, datastore=None, os="rhel6_64Guest", enter_bios=False):...
```

```python
# -*- coding: utf-8 -*-
import logging

logger = logging.getLogger()

from pysphere.vi_virtual_machine import *
from pysphere.resources.vi_exception import *

from ESXiHost import *


class ESXiGuestClass: #用户操作类
    def __init__(self, esxihost, name):...
    #快照方法
    def create_snapshot(self):...
    #删除快照
    def delete_snapshot(self):...
    #覆盖快照方法
    def revert_to_snapshot(self):...
    #返回快照信息
    def get_snapshot_info(self):...
    #返回状态
    def get_status(self):...
    #开机
    def power_on(self):...
    #关机
    def power_off(self):...
    #重启
    def reboot(self):...

    def blalala(self):...

    def remove_me(self):...

    def set_enter_bios(self):...

    def create_me(self, cpu=1, mem=1024, network=None, diskGB=2, datastore=None, os="rhel6_64Guest", enter_bios=False):...
```

运行结果

可运行的命令行

show list-host and list-guest

python ./pySimpleVmCtrl.py -H 192.168.221.128 -U root -P yekaikai,./123 -A list-host -A list-guest

turn on/off/reboot guest

python ./pySimpleVmCtrl.py -H 192.168.221.128 -U root -P yekaikai,./123 -A on -g test-01 python ./pySimpleVmCtrl.py -H 192.168.221.128 -U root -P yekaikai,./123 -A off -g test-01 python ./pySimpleVmCtrl.py -H 192.168.221.128 -U root -P yekaikai,./123 -A reboot -g test-01

create guest

python ./pySimpleVmCtrl.py -H 192.168.221.128 -U root -P yekaikai,./123 -A create --disk 10 --store "[datastore1]" --cpu 1 --mem 2048 --net LAN -g test-02

get_snapshot_info

python ./pySimpleVmCtrl.py -H 192.168.221.128 -U root -P yekaikai,./123 -A get_snapshot_info -g test-01

create_snapshot

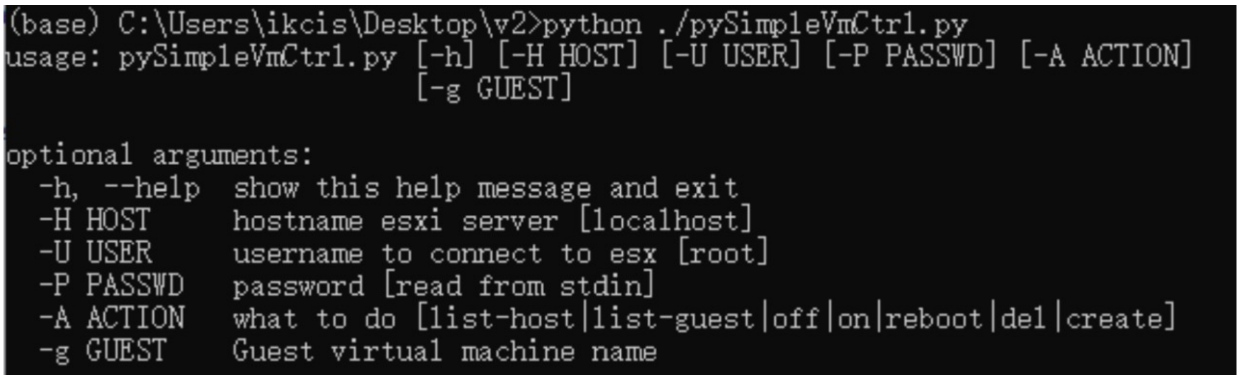
python ./pySimpleVmCtrl.py -H 192.168.221.128 -U root -P yekaikai,./123 -A create_snapshot -g test-01

revert_to_snapshot

python ./pySimpleVmCtrl.py -H 192.168.221.128 -U root -P yekaikai,./123 -A revert_to_snapshot -g test-01

运行截图

Usage&Argument

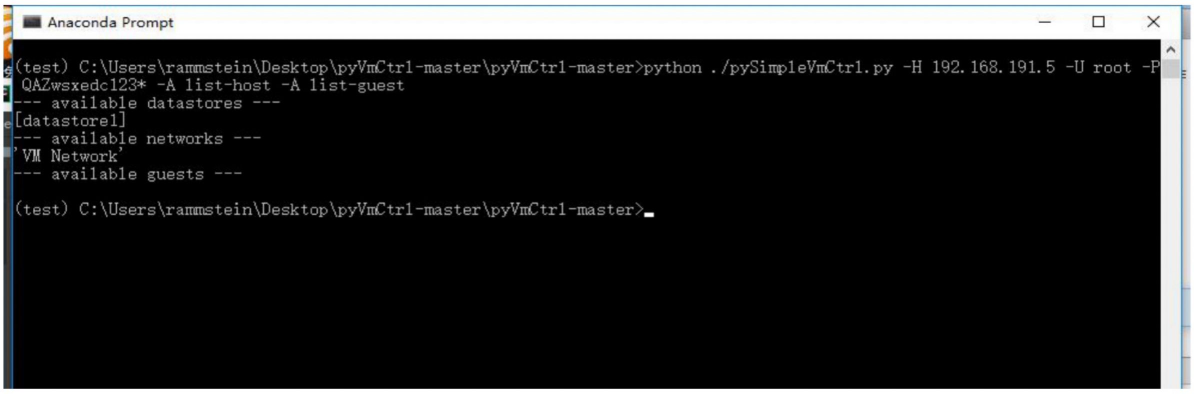


```
(base) C:\Users\ikcis\Desktop\v2>python ./pySimpleVmCtrl.py
usage: pySimpleVmCtrl.py [-h] [-H HOST] [-U USER] [-P PASSWD] [-A ACTION]
                         [-g GUEST]

optional arguments:
  -h, --help  show this help message and exit
  -H HOST     hostname esxi server [localhost]
  -U USER     username to connect to esx [root]
  -P PASSWD   password [read from stdin]
  -A ACTION   what to do [list-host|list-guest|off|on|reboot|del|create]
  -g GUEST    Guest virtual machine name
```

测试举例

show list-host and list-guest

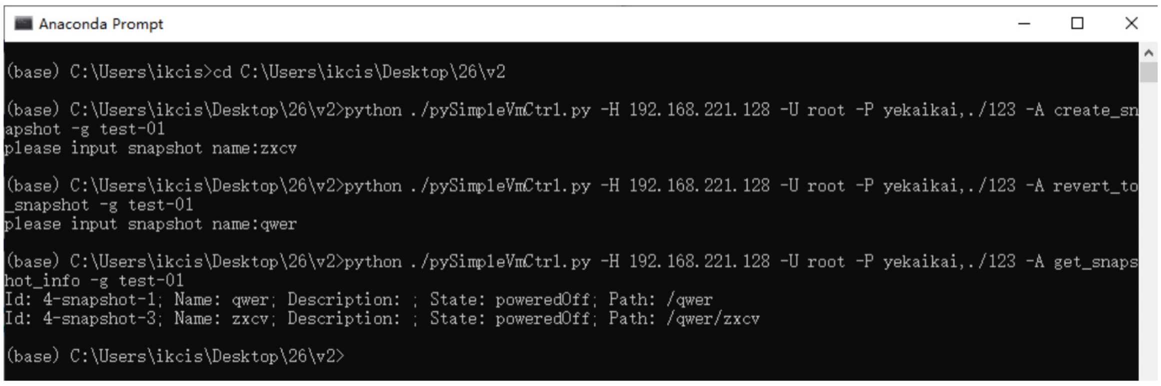
python ./pySimpleVmCtrl.py -H 192.168.221.128 -U root -P yekaikai,./123 -A list-host -A list-guest



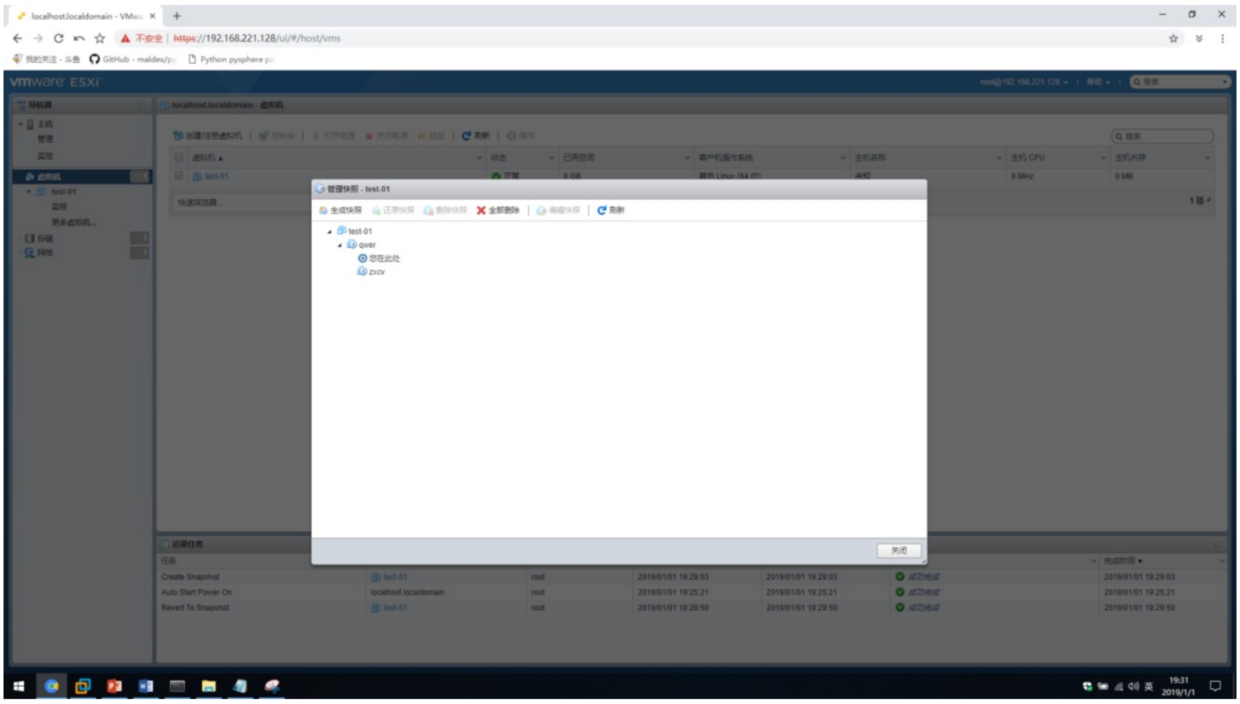
```
(test) C:\Users\rammstein\Desktop\pyVmCtrl-master\pyVmCtrl-master>python ./pySimpleVmCtrl.py -H 192.168.191.5 -U root -P
 QAZwsxedc123* -A list-host -A list-guest
--- available datastores ---
[datastore1]
--- available networks ---
'VM Network'
--- available guests ---

(test) C:\Users\rammstein\Desktop\pyVmCtrl-master\pyVmCtrl-master>_
```

get_snapshot_info

python ./pySimpleVmCtrl.py -H 192.168.221.128 -U root -P yekaikai,./123 -A get_snapshot_info -g test-01



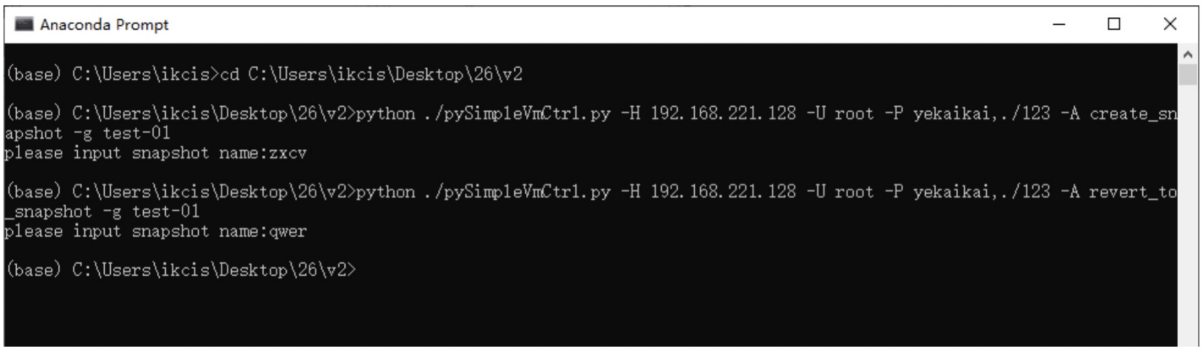
```
(base) C:\Users\ikcis>cd C:\Users\ikcis\Desktop\26\v2

(base) C:\Users\ikcis\Desktop\26\v2>python ./pySimpleVmCtrl.py -H 192.168.221.128 -U root -P yekaikai,./123 -A create_sn
apshot -g test-01
please input snapshot name:zxcv

(base) C:\Users\ikcis\Desktop\26\v2>python ./pySimpleVmCtrl.py -H 192.168.221.128 -U root -P yekaikai,./123 -A revert_to
_snapshot -g test-01
please input snapshot name:qwer

(base) C:\Users\ikcis\Desktop\26\v2>python ./pySimpleVmCtrl.py -H 192.168.221.128 -U root -P yekaikai,./123 -A get_snaps
hot_info -g test-01
Id: 4-snapshot-1; Name: qwer; Description: ; State: poweredOff; Path: /qwer
Id: 4-snapshot-3; Name: zxcv; Description: ; State: poweredOff; Path: /qwer/zxcv

(base) C:\Users\ikcis\Desktop\26\v2>
```



```
Create Snapshot      test-01               root   2019/01/01 19:29:03   2019/01/01 19:29:03   成功完成   2019/01/01 19:29:03
Auto Start Power On  localhost.localdomain root   2019/01/01 19:25:21   2019/01/01 19:25:21   成功完成   2019/01/01 19:25:21
Revert To Snapshot   test-01               root   2019/01/01 19:29:50   2019/01/01 19:29:50   成功完成   2019/01/01 19:29:50
```

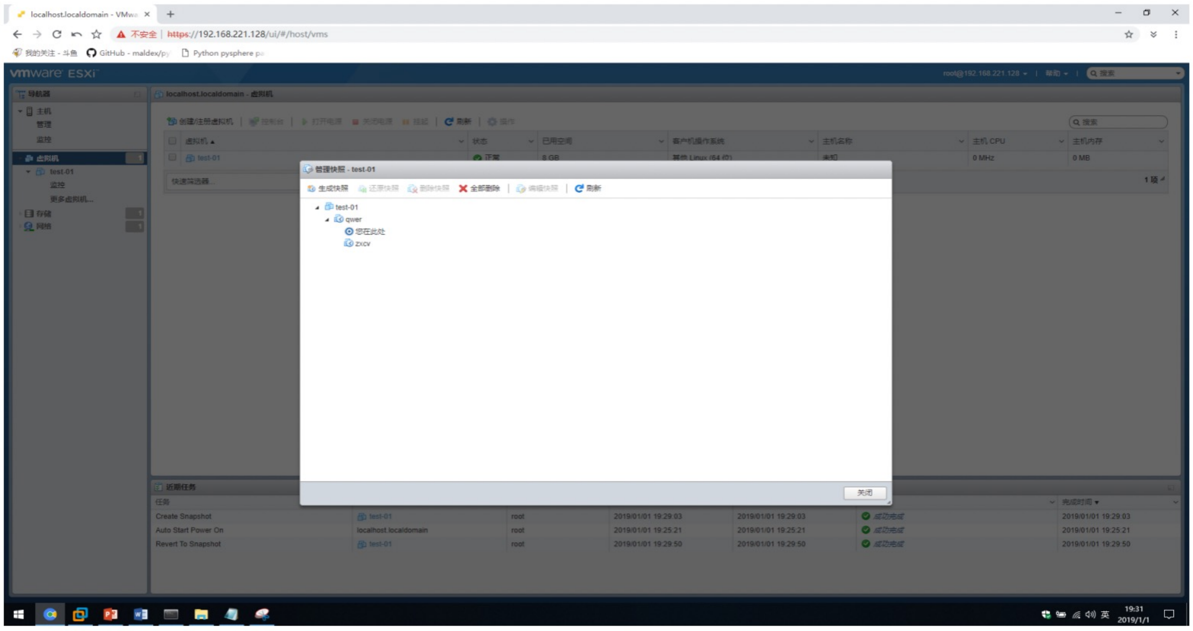
revert_to_snapshot

python ./pySimpleVmCtrl.py -H 192.168.221.128 -U root -P yekaikai,./123 -A revert_to_snapshot -g test-01



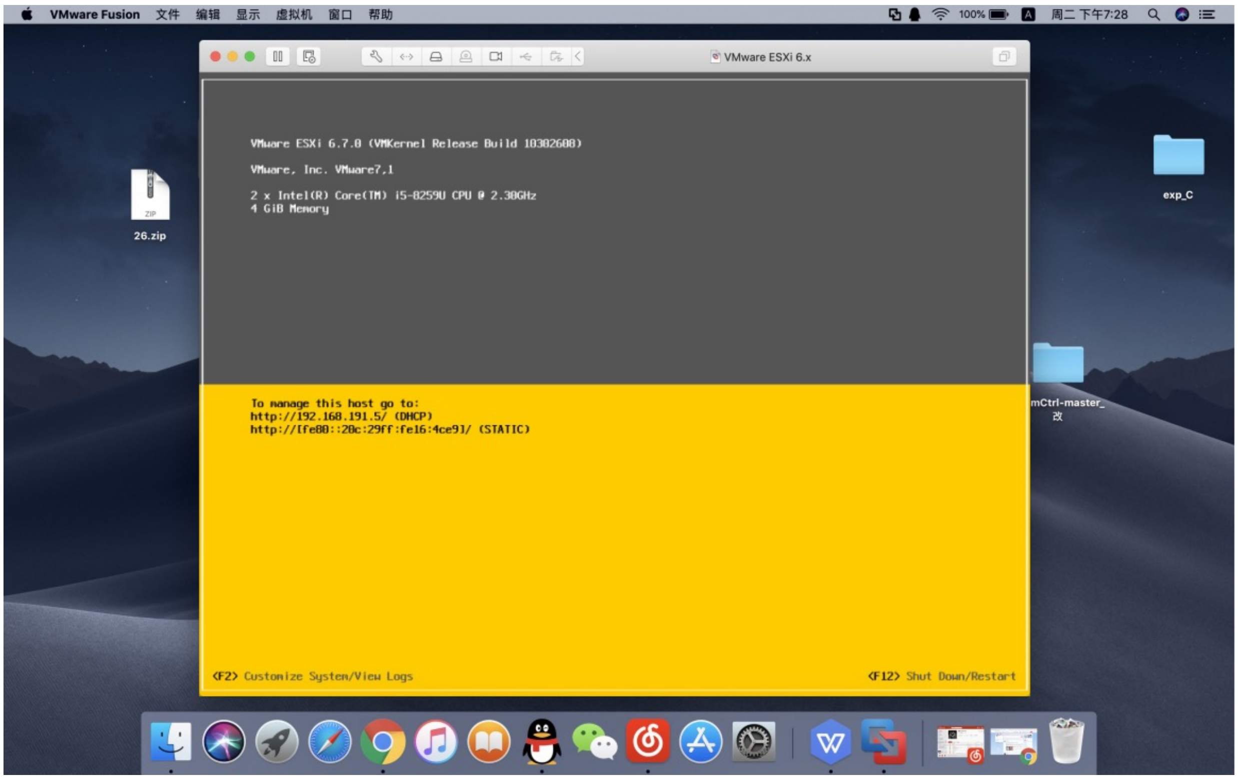
```
(base) C:\Users\ikcis>cd C:\Users\ikcis\Desktop\26\v2

(base) C:\Users\ikcis\Desktop\26\v2>python ./pySimpleVmCtrl.py -H 192.168.221.128 -U root -P yekaikai,./123 -A create_sn
apshot -g test-01
please input snapshot name:zxcv

(base) C:\Users\ikcis\Desktop\26\v2>python ./pySimpleVmCtrl.py -H 192.168.221.128 -U root -P yekaikai,./123 -A revert_to
_snapshot -g test-01
please input snapshot name:qwer

(base) C:\Users\ikcis\Desktop\26\v2>
```

与Web后台界面信息相同，验证。

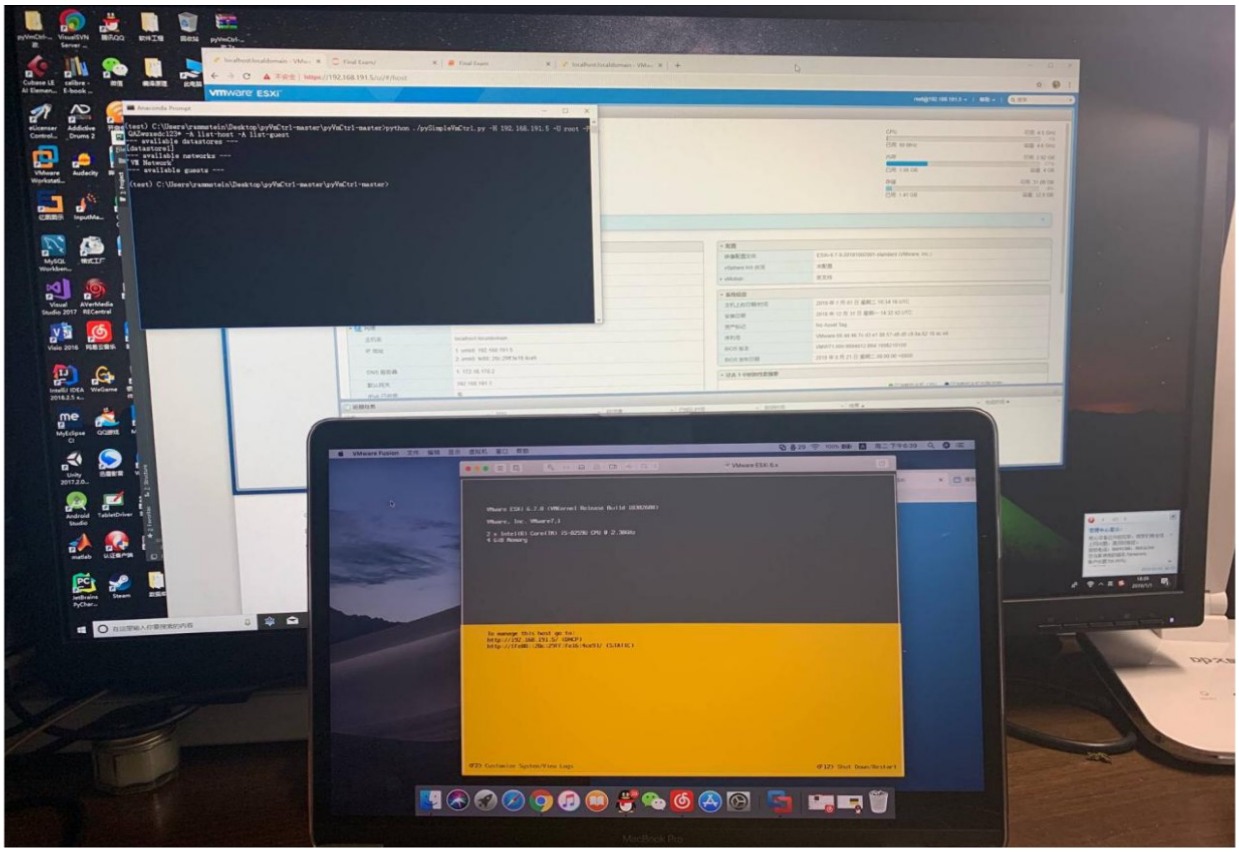


任务						完成时间
Create Snapshot	test-01	root	2019/01/01 19:29:03	2019/01/01 19:29:03	成功完成	2019/01/01 19:29:03
Auto Start Power On	localhost.localdomain	root	2019/01/01 19:25:21	2019/01/01 19:25:21	成功完成	2019/01/01 19:25:21
Revert To Snapshot	test-01	root	2019/01/01 19:29:50	2019/01/01 19:29:50	成功完成	2019/01/01 19:29:50

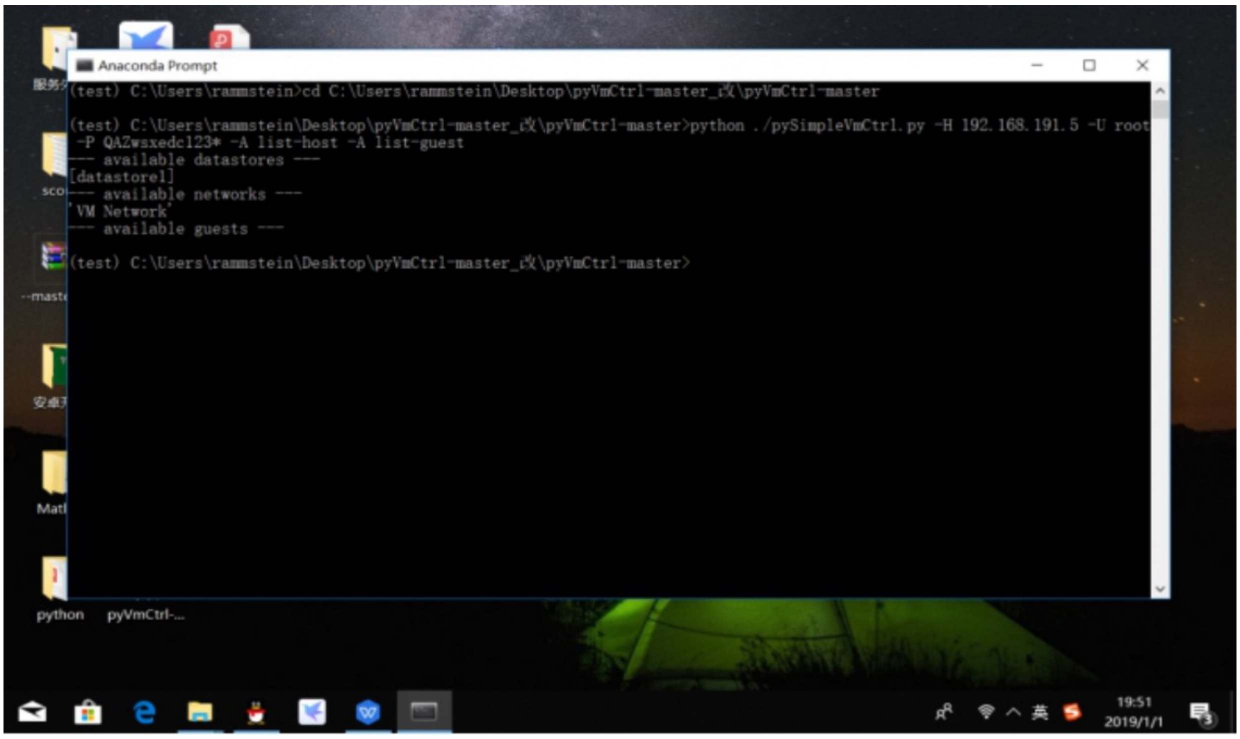
远程连接服务器



```
VMware ESXi 6.7.0 (VMKernel Release Build 10302608)

VMware, Inc. VMware7,1

2 x Intel(R) Core(TM) i5-8259U CPU @ 2.30GHz
4 GiB Memory

To manage this host go to:
http://192.168.191.5/ (DHCP)
http://[fe80::20c:29ff:fe16:4ce9]/ (STATIC)

<F2> Customize System/View Logs                                  <F12> Shut Down/Restart
```



```
(test) C:\Users\rammstein\Desktop\pyVmCtrl-master\pyVmCtrl-master>python ./pySimpleVmCtrl.py -H 192.168.191.5 -U root -P
QAZwsxedc123* -A list-host -A list-guest
--- available datastores ---
[datastore1]
--- available networks ---
'VM Network'
--- available guests ---

(test) C:\Users\rammstein\Desktop\pyVmCtrl-master\pyVmCtrl-master>
```



```
(test) C:\Users\rammstein>cd C:\Users\rammstein\Desktop\pyVmCtrl-master_改\pyVmCtrl-master

(test) C:\Users\rammstein\Desktop\pyVmCtrl-master_改\pyVmCtrl-master>python ./pySimpleVmCtrl.py -H 192.168.191.5 -U root
 -P QAZwsxedc123* -A list-host -A list-guest
--- available datastores ---
[datastore1]
--- available networks ---
'VM Network'
--- available guests ---

(test) C:\Users\rammstein\Desktop\pyVmCtrl-master_改\pyVmCtrl-master>
```

自我评价

1、PySphere库提供了很多针对虚拟机操作的方法，需要仔细查阅官方说明文档，并在程序中正确调用这些方法并且捕获在调用过程中可能抛出的异常。

2、 程序的难点在于ESXi服务端后台类的编写，因为对于ESXiHostClass，在PySphere细分了很多调用方法且要通过SSL进行连接访问。

3、 命令行对参数的控制更加浅显易懂，但设计成脚本的形式可能更方便。

4、没有完成的功能：虚拟机克隆、图形化界面

小组分工

1、ESXi服务端后台类的编写，包括初始化、建立连接和实现虚拟机的创建和删除。（吴俚达）

2、ESXi虚拟机类的编写，包括初始化、开机关机重启、快照的创建删除回滚查询操作等功能。（叶凯凯）

3、用argparse库将程序转换成命令行解析程序，实现通过命令行参数的组合来控制程序的运行，同时可以产生帮助信息。（葛华盛）

4、PPT制作（叶凯凯）、文档制作（吴俚达）

附录

In [ ]:

```python
# ESXiHost.py

import ssl

default_context = ssl._create_default_https_context

import logging

logger = logging.getLogger()

from pysphere import VIServer, VIProperty
from pysphere.resources import VimService_services as VI
from pysphere.vi_mor import VIMor


class ESXiHostClass:
    def __init__(self, host, user, passwd):
        self._connection = VIServer()
        logger.info("%s:connecting to '%s' as '%s'", __name__, host, user)
        try:
            ssl._create_default_https_context = ssl._create_unverified_context
            self._connection.connect(host, user, passwd)
            logger.debug("%s:host reports '%s V.%s'", __name__,
                         self._connection.get_server_type(), self._connection.get_api_version())
            self.host_config, n, n, n = self._get_host_config()
        except Exception, err:
            logger.critical("%s:%s", __name__, err)
            quit(2)

    def get_connection(self):
        return self._connection

    def get_guests(self):
        ret = []
        for each in self._connection.get_registered_vms():
            entry = self._connection.get_vm_by_path(each)
            ret.append(entry.get_properties()['name'])
        logger.debug("%s:found %s guests", __name__, len(ret))
        return ret

    def _get_host_config(self):
        # -> get Datacenter and it's properties
        dc_mor = [k for k, v in self._connection.get_datacenters().items()][-1]  # just take the l
        dc_props = VIProperty(self._connection, dc_mor)

        # -> get computer resources MORs
        cr_mors = self._connection._retrieve_properties_traversal(property_names=['name', 'host'],
                                                                  from_node=dc_props.hostFolder._ol
                                                                  obj_type='ComputeResource')

        # -> get host MOR
        host_mor = [k for k, v in self._connection.get_hosts().items()][-1]  # just take the last

        # -> get computer resource MOR for host
        cr_mor = None
        for cr in cr_mors:
            if cr_mor:
                break
            for p in cr.PropSet:
                if p.Name == "host":
                    for h in p.Val.get_element_ManagedObjectReference():
                        if h == host_mor:
                            cr_mor = cr.Obj
                            break
                    if cr_mor:
                        break
        # -> get Computer Ressources
        cr_props = VIProperty(self._connection, cr_mor)

        # -> create configuration request()
        request = VI.QueryConfigTargetRequestMsg()
        _this = request.new__this(cr_props.environmentBrowser._obj)
        _this.set_attribute_type(cr_props.environmentBrowser._obj.get_attribute_type())
        request.set_element__this(_this)

        # -> get answer back!
        config_target = self._connection._proxy.QueryConfigTarget(request)._returnval
        return config_target, host_mor, dc_props, cr_props

    def get_networks(self):
        ret = []
        for net in self.host_config.Network:
            ret.append(net.Network.Name)
        logger.debug("%s:found %s networks", __name__, len(ret))
        return ret

    def get_datastores(self):
        ret = []
        for d in self.host_config.Datastore:
            if d.Datastore.Accessible:
                ret.append(d.Datastore.Name)
        logger.debug("%s:found %s datastores", __name__, len(ret))
        return ret
```

In [ ]:
```python
# ESXiGuest.py

import logging

logger = logging.getLogger()

from pysphere.vi_virtual_machine import *
from pysphere.resources.vi_exception import *

from ESXiHost import *


class ESXiGuestClass:
    def __init__(self, esxihost, name):
        assert isinstance(esxihost, ESXiHostClass)
        self._host = esxihost
        self.name = name
        try:
            self.vm = self._host.get_connection().get_vm_by_name(name)
            logger.debug("%s:found guest named '%s'", __name__, name)
            self.config = self.vm.get_properties()
        except VIException, e:
            logger.warn("%s:%s", __name__, e.message)
            self.vm = None

    def create_snapshot(self):
        assert isinstance(self.vm, VIVirtualMachine)
        snapshot_name = raw_input("please input snapshot name:")
        self.vm.create_snapshot(name=snapshot_name)

    def delete_snapshot(self):
        assert isinstance(self.vm, VIVirtualMachine)
        snapshot_name = raw_input("please input snapshot name:")
        self.vm.delete_named_snapshot(name=snapshot_name)

    def revert_to_snapshot(self):
        assert isinstance(self.vm, VIVirtualMachine)
        snapshot_name = raw_input("please input snapshot name:")
        self.vm.revert_to_named_snapshot(name=snapshot_name)

    def get_snapshot_info(self):
        assert isinstance(self.vm, VIVirtualMachine)
        snapshot_list = self.vm.get_snapshots()
        for snapshot in snapshot_list:
            snaptext = 'Id: %s; Name: %s; Description: %s; State: %s; Path: %s' % (
                snapshot._mor, snapshot.get_name(), snapshot.get_description(),
                snapshot.get_state(), snapshot.get_path())
            print snaptext

    def get_status(self):
        assert isinstance(self.vm, VIVirtualMachine)
        netstat = 'N/A'
        ns = self.vm.get_property('net', from_cache=False)
        if ns is not None:
            netstat = ns[0]['network']
            netstat += '=' + ns[0]['mac_address']
            try:
                netstat += '=' + str(ns[0]['ip_addresses'][0])
            except IndexError:
                pass

        return self.vm.get_status(), netstat

    def power_on(self):
        assert isinstance(self.vm, VIVirtualMachine)
        try:
            self.vm.power_on()
            return True
        except VIException, e:
            logger.error("%s:'%s':%s ", __name__, self.name, e.message)
        return False

    def power_off(self):
        assert isinstance(self.vm, VIVirtualMachine)
        try:
            self.vm.shutdown_guest()
            return True
        except VIApiException, e:
            logger.error("%s:'%s':failed to shutdown, KILLING GUEST! %s ", __name__, self.name, e.
        except TypeError, e:
            logger.critical("%s:'%s':API ERROR?! KILLING GUEST! %s ", __name__, self.name, str(e))
        try:
            self.vm.power_off()
            return True
        except VIException, e:
            logger.error("%s:'%s':%s ", __name__, self.name, e.message)
        return False

    def reboot(self):
        assert isinstance(self.vm, VIVirtualMachine)
        try:
            self.vm.reboot_guest()
            return True
        except VIApiException, e:
            logger.error("%s:'%s':failed to reboot, POWER-CYCLING GUEST! %s ", __name__, self.name,
        except TypeError, e:
            logger.critical("%s:'%s':API ERROR?! KILLING GUEST! %s ", __name__, self.name, str(e))
        try:
            self.power_off()
            return self.power_on()
        except VIException, e:
            logger.error("%s:'%s':%s ", __name__, self.name, e.message)
        return False

    def remove_me(self):
        remove_vm_request = VI.Destroy_TaskRequestMsg()
        _this = remove_vm_request.new__this(self.vm._mor)
        _this.set_attribute_type(self.vm._mor.get_attribute_type())
        remove_vm_request.set_element__this(_this)
        ret = self._host._connection._proxy.Destroy_Task(remove_vm_request)._returnval

        # Wait for the task to finish
        task = VITask(ret, self._host._connection)

        status = task.wait_for_state([task.STATE_SUCCESS, task.STATE_ERROR])
        if status == task.STATE_SUCCESS:
            print "VM successfully deleted from disk"
        elif status == task.STATE_ERROR:
            print "Error removing vm:", task.get_error_message()

    def create_me(self, cpu=1, mem=1024, network=None, diskGB=2, datastore=None, os="rhel6_64Guest

        if not network:
            network = self._host.get_networks()[-1]

        if not datastore:
            datastore = self._host.get_datastores()[-1]

        if not datastore.startswith('[') or not datastore.endswith(']'):
            datastore = '[' + datastore + ']'

        #
        #  prepare a VM Creation request
        #
        create_vm_request = VI.CreateVM_TaskRequestMsg()
        config = create_vm_request.new_config()

        # set the datastore path
        vm_files = config.new_files()
        vm_files.set_element_vmPathName(datastore)
        config.set_element_files(vm_files)

        # set some basics
        config.set_element_version("vmx-08")
        config.set_element_name(self.name)
        config.set_element_memoryMB(mem)
        config.set_element_memoryHotAddEnabled(True)
        config.set_element_numCPUs(cpu)
        config.set_element_cpuHotAddEnabled(True)
        config.set_element_guestId(os)

        if enter_bios:
            # set boot parameters
            vmboot = config.new_bootOptions()
            vmboot.set_element_enterBIOSSetup(True)
            config.set_element_bootOptions(vmboot)

        # -> add devices
        devices = []

        # add network interface
        if network:
            nic_spec = config.new_deviceChange()
            nic_spec.set_element_operation("add")
            nic_ctlr = VI.ns0.VirtualVmxnet3_Def("nic_ctlr").pyclass()
            nic_backing = VI.ns0.VirtualEthernetCardNetworkBackingInfo_Def("nic_backing").pyclass(
            nic_backing.set_element_deviceName(network)
            nic_ctlr.set_element_addressType("generated")
            nic_ctlr.set_element_backing(nic_backing)
            nic_ctlr.set_element_key(4)
            nic_spec.set_element_device(nic_ctlr)
            devices.append(nic_spec)

        if diskGB:
            # add controller to devices
            disk_ctrl_key = 1
            scsi_ctrl_spec = config.new_deviceChange()
            scsi_ctrl_spec.set_element_operation('add')
            scsi_ctrl = VI.ns0.ParaVirtualSCSIController_Def("scsi_ctrl").pyclass()

            scsi_ctrl.set_element_busNumber(0)
            scsi_ctrl.set_element_key(disk_ctrl_key)
            scsi_ctrl.set_element_sharedBus("noSharing")
            scsi_ctrl_spec.set_element_device(scsi_ctrl)
            devices.append(scsi_ctrl_spec)

            # add disk
            disk_spec = config.new_deviceChange()
            disk_spec.set_element_fileOperation("create")
            disk_spec.set_element_operation("add")
            disk_ctlr = VI.ns0.VirtualDisk_Def("disk_ctlr").pyclass()
            disk_backing = VI.ns0.VirtualDiskFlatVer2BackingInfo_Def("disk_backing").pyclass()
            disk_backing.set_element_fileName(datastore)
            disk_backing.set_element_diskMode("persistent")
            disk_ctlr.set_element_key(0)
            disk_ctlr.set_element_controllerKey(disk_ctrl_key)
            disk_ctlr.set_element_unitNumber(3)
            disk_ctlr.set_element_backing(disk_backing)
            guest_disk_size = diskGB * 1024 * 1024
            disk_ctlr.set_element_capacityInKB(guest_disk_size)
            disk_spec.set_element_device(disk_ctlr)
            devices.append(disk_spec)
        # place where the new Guest shall shall go to
        config_target, host_mor, dc_props, cr_props = self._host._get_host_config()
        vmf_mor = dc_props.vmFolder._obj  # get vmFolder
        rp_mor = cr_props.resourcePool._obj  # get resource pool MOR

        # append devices to the new config
        config.set_element_deviceChange(devices)

        # assemble the request
        create_vm_request.set_element_config(config)
        new_vmf_mor = create_vm_request.new__this(vmf_mor)
        new_vmf_mor.set_attribute_type(vmf_mor.get_attribute_type())
        new_rp_mor = create_vm_request.new_pool(rp_mor)
        new_rp_mor.set_attribute_type(rp_mor.get_attribute_type())
        new_host_mor = create_vm_request.new_host(host_mor)
        new_host_mor.set_attribute_type(host_mor.get_attribute_type())
        create_vm_request.set_element__this(new_vmf_mor)
        create_vm_request.set_element_pool(new_rp_mor)
        create_vm_request.set_element_host(new_host_mor)

        # finally actually create the guest :)
        task_mor = self._host._connection._proxy.CreateVM_Task(create_vm_request)._returnval
        task = VITask(task_mor, self._host._connection)
        task.wait_for_state([task.STATE_SUCCESS, task.STATE_ERROR])

        if task.get_state() == task.STATE_ERROR:
            print "Cannot create guest: ", task.get_info(), task.get_error_message()
        else:
            print "Succesfully created guest! "
            self.__init__(self._host, self.name)
```

In [ ]:

```python
# pySimpleVmCtrl.py

import argparse
from pySimpleVmCtrl.ESXiGuest import *

logger = logging.getLogger()

parser = argparse.ArgumentParser(formatter_class=argparse.RawTextHelpFormatter)

parser.add_argument('-v', action='store_true', default=False,
                    dest='verbosity',
                    help='be verbose [%(default)s]')

parser.add_argument('-H', action='store', default='localhost',
                    dest='host',
                    help='hostname esxi server [%(default)s]')

parser.add_argument('-U', action='store', default='root',
                    dest='user',
                    help='username to connect to esx [%(default)s]')

parser.add_argument('-P', action='store',
                    dest='passwd',
                    help='password [read from stdin]')

parser.add_argument('-A', action='append', dest='action',
                    help="what to do [list-host|list-guest|off|on|reboot|del|create]",
                    )

parser.add_argument('-g', action='store',
                    dest='guest',
                    help='Guest virtual machine name')

parser.add_argument('--store', action='store', default=None,
                    dest='datastore',
                    help='(create) datastore to use')

parser.add_argument('--net', action='store', default=None,
                    dest='network',
                    help='(create) network to connect to')

parser.add_argument('--disk', action='store', default=8,
                    dest='disksize',
                    help='(create) disksize in GB [%(default)s]')

parser.add_argument('--cpu', action='store', default="1",
                    dest='cpu',
                    help='(create) cpu count [%(default)s]')

parser.add_argument('--mem', action='store', default="1024",
                    dest='memory',
                    help='(create) memory in MB [%(default)s]')

parser.add_argument('--os', action='store', default="rhel6_64Guest",
                    dest='operatingsystem',
                    help='(create) Operating system [%(default)s]')


def execute_arguments(esx_host, action, args):
    logger.debug('%s:execute_arguments(%s, %s, %s)', __name__, 'ESXiHostClass', action, args.guest)
    if action == "list-host":
        print "--- available datastores ---"
        for each in esx_host.get_datastores():
            print "[" + each + "]"
        print "--- available networks ---"
        for each in esx_host.get_networks():
            print "'" + each + "'"
        return True

    elif action == "list-guest":
        print "--- available guests ---"
        for each in esx_host.get_guests():
            each_guest = ESXiGuestClass(esx_host, each)
            power, net = each_guest.get_status()
            print each, ' ', power, ' ', net
        return True

    if args.guest is None:
        logger.critical('%s:execute_arguments(): either illegal action or specify guest name', __na
        return False

    guest = ESXiGuestClass(esx_host, args.guest)

    assert isinstance(guest, ESXiGuestClass)

    if action == "off":
        return guest.power_off()
    if action == "on":
        return guest.power_on()
    if action in ["reset", "reboot"]:
        return guest.reboot()

    if action in ["add", "create"]:
        return guest.create_me(cpu=int(argparser.cpu), mem=int(argparser.memory), os=argparser.ope
                               datastore=argparser.datastore, network=argparser.network, diskGB=int

    if action in ["rem", "del"]:
        guest.power_off()
        return guest.remove_me()

    if action in ["create_snapshot"]:
        return guest.create_snapshot()

    if action in ["delete_snapshot"]:
        return guest.delete_snapshot()

    if action in ["revert_to_snapshot"]:
        return guest.revert_to_snapshot()

    if action in ["get_snapshot_info"]:
        return guest.get_snapshot_info()


if __name__ == "__main__":
    argparser = parser.parse_args()

    if argparser.action is None:
        parser.print_help()
        quit(2)

    esx_host = ESXiHostClass(argparser.host, argparser.user, argparser.passwd)

    for each_action in argparser.action:
        execute_arguments(esx_host, each_action, argparser)
```

In [ ]: