

به نام ایزد بخشاینده مهربان



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

گزارش تمرین اول

درس مبانی رایانش ابری

استاد: دکتر سید احمد جوادی

اعضای گروه:

علی شفیعی - ۹۵۳۱۸۰۱

سید سجاد پیشوائیان - ۹۵۳۱۰۱۵

آبان ۱۳۹۹

لازم به ذکر است که تمامی بخش‌ها با همکاری هر دو عضو با استفاده از skype زده شده است و کدینگ کاملاً در هر بخش دو نفره و اشتراکی بوده است

بخش اول:

۱-۱- **Oracle VM VirtualBox**: یک نرم افزار مجازی سازی دسکتاپ cross-platform قابل حمل و کامل است که به شما امکان می دهد چندین سیستم عامل را به طور همزمان اجرا کنید. با این کار می توانید ماشین های مجازی زنده را از یک میزبان یا ابر به میزبان دیگر انتقال داد بدون اینکه روند کار باعث قطع عملکرد آن شود. به این ترتیب ، در مورد قرارگیری ماشین های مجازی خود حداکثر انعطاف را دارید. همچنین در مواردی که به فضای پیچیده مجازی بزرگ و فضای ذخیره سازی پیشرفته نیاز نیست ، می تواند بسیار خوب کار کند.

VirtualBox همچنین به شما امکان می دهد از ماشین های مجازی خود snapshot بگیرید. این به شما امکان می دهد در صورت مواجه شدن با مشکل ، وضعیت فعلی آنها را برای بازسازی در آینده ذخیره کنید. جدا از آن ، می توانید از عکسهای فوری برای جابجایی بین تاریخها استفاده کنید ، و نوعی واقعیت متناوب ایجاد کنید ، بنابراین می توانید به راحتی ماشینهای خود را به روشهای مختلف پیکربندی کنید.

علاوه بر این ، VirtualBox دارای احراز هویت RDP قابل توسعه است. علاوه بر پشتیبانی از Winlogon ، دارای یک SDK است که می توانید از آن برای ایجاد سایر مراحل احراز هویت استفاده کنید. وقتی گزینه های مختلفی دارید ، می توانید برای امنیت بیشتر دسترسی به سیستم خود را سفارشی کنید.

به طور کلی ، VirtualBox به یک سیستم عامل میزبان نیاز دارد ، بنابراین کاملاً مانند یک سرور مجازی واقعی headless کار نخواهد کرد. بنابراین نباید انتظار داشت که این به راه حلی برای جایگزینی مجموعه ای از سرورها با یک فضای مجازی گسترده ، خودکار و اضافی تبدیل شود.

مزایا:

- می تواند چندین ماشین مجازی را در یک شبکه host-only بچرخاند که با یکدیگر صحبت می کنند و امکان ایجاد محیط های آزمایشی جذاب را فراهم می کند.
- قابلیت Cross-platform که آموزش و یادگیری آن در محیط های ویندوز و لینوکس و سیستم عامل OSX را آسان می سازد.
- VirtualBox در هنگام سازگاری بالاترین میزان پشتیبانی و مستند سازی را دارد.

معایب:

- مصرف حافظه بالا (مثلاً برای اجرای روان Ubuntu VM به بیش از ۸ گیگابایت حافظه نیاز دارد).
- برخی از مشکلات شناخته شده در زمینه imprt/export تصاویر OVA / OVF. این مسئله وجود دارد که برخی بخشها در OVF نادیده گرفته می شوند (خصوصاً بخشهای Install و Startup).
- برخی از مشکلات متناوب هنگام نصب دیسک های سخت خارجی و تلاش برای دسترسی به آنها از داخل VM در حال اجرا.

Xen: پروژه Xen یک hypervisor بدون پوشش منبع باز است که یک پروژه مشترک توسط بنیاد لینوکس و مشارکت کنندگان شریک است. این پروژه به ویژگی های قدرتمندی مجهز است که با نرم افزارهای تجاری در رقابت هستند و می توانید از آنها برای تأمین قدرت سازمان خود استفاده کنید.

پروژه Xen همچنین به اصل خود یعنی واژه یونانی باستان Xenos که به معنی مهمان دوستان است ، پابرجا می ماند. به این دلیل که می تواند از چندین سیستم عامل مهمان پشتیبانی کند تا انعطاف پذیری لازم برای افزودن سیستم عامل های بیشتر بسته به نیاز شما را فراهم کند.

علاوه بر پشتیبانی از سیستم عامل چند میهمان ، Xen Project می تواند با سیستم عامل های ابری نیز کار کند. این به شما آزادی می دهد که گزینه های خود را باز نگه دارید تا بتوانید رشد و بهبود راه حل های ابری را به جای سپردن و قفل شدن برای مدت طولانی ، مشاهده کنید.

مزایا:

- از طریق کنسول (در صورت مجوز) با سهولت نسبی ، به روزرسانی های مجموعه میزبان های hypervisor را مدیریت می کند.
- ایجاد الگوها و clone های ماشین و جابجایی آنها را در داخل میزبان آسان می کند.
- با استفاده از پایگاه داده کپی شده روی هر میزبان ، به خوبی خطا ها را مدیریت می کند.

معایب:

- پشتیبانی - اگر رایگان بودن همان چیزی است که شما نیاز دارید ، پشتیبانی عالی آن چیزی نیست که دریافت خواهد شد.
- ارتقا - روند ارتقا در بهترین حالت مشکل دار و مشکل ساز است.

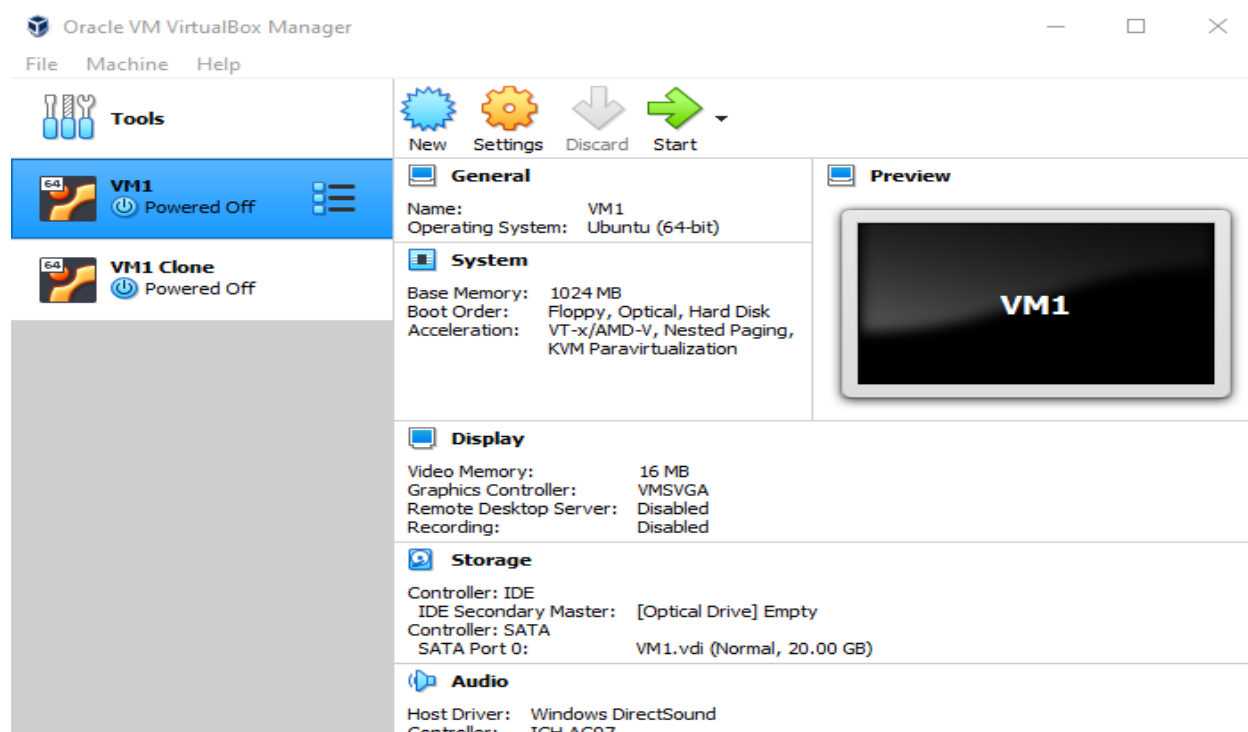
۲-۱- ویژگی ها و معیار های کلیدی:

Oracle VM VirtualBox	Xen
Cross-Platform Virtualization	Multiple Guest Operating Systems
Guest Multiprocessing	Xen Security Modules
ACPI Support	Virtual Machine Migration
Branched Snapshots	Multi-Vendor Support
Remote Machine Display	Multiple Cloud Platforms Support
Guest Additions	Flexible & Modular Architecture
USB Device Support	Open Source
Multiscreen Resolutions	
PXE Network Boot	
VM Groups	
Extensible RDP Authentication	
Built-In iSCSI Support	

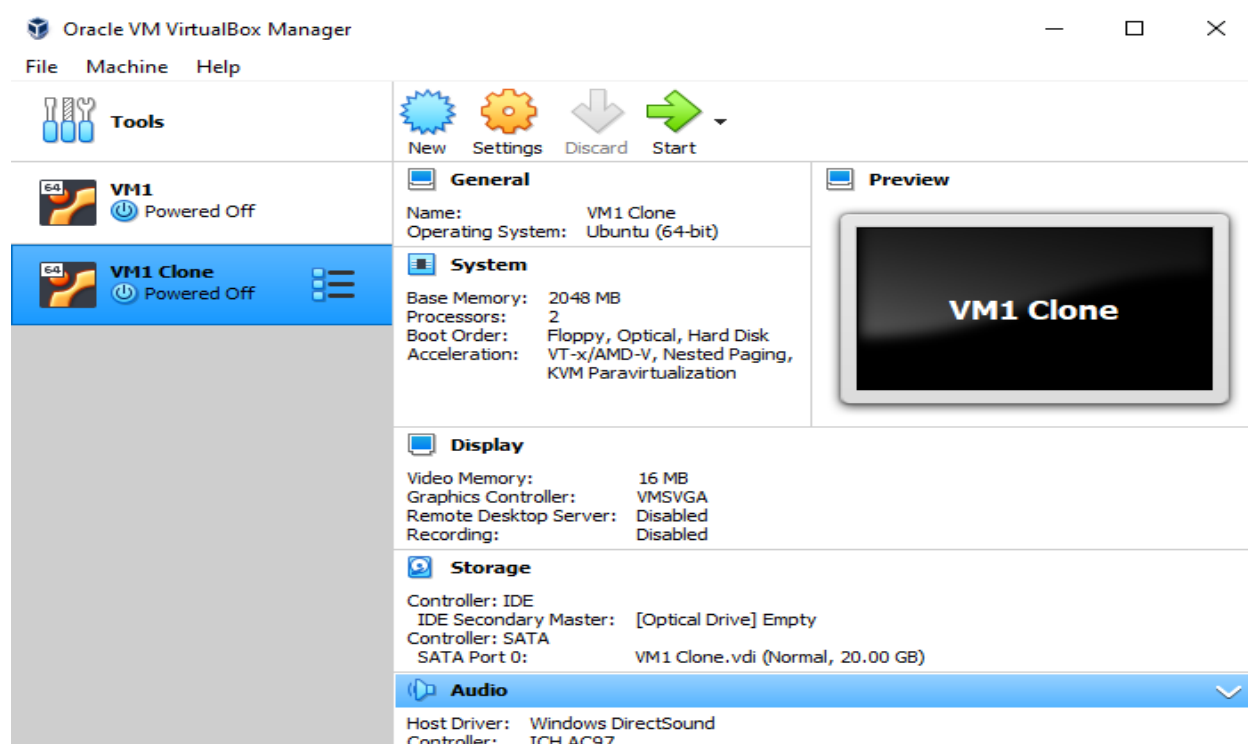
بخش دوم:

در این قسمت ما برای پیاده سازی موارد خواسته شده از زبان برنامه نویسی پایتون استفاده کرده ایم که هم دارای رابط گرافیکی است و هم می توان آن را روی سرور Jupyter اجرا نمود(هر دو برنامه در فایل آپلودی موجود است). ما مراحل نصب و پیکربندی VirtualBox را روی محیط ویندوز انجام داده ایم و همانطور که در صورت تمرین آمده بود پیکربندی نموده ایم. همچنین برای کار با VirtualBox از طریق رابط کاربری از کتابخانه pyvbox استفاده نموده ایم که کتابخانه اختصاصی کار با VirtualBox برای زبان برنامه نویسی پایتون می باشد. در ادامه ما عکس ها و کد های مربوط به هر بخش را توضیح می دهیم.

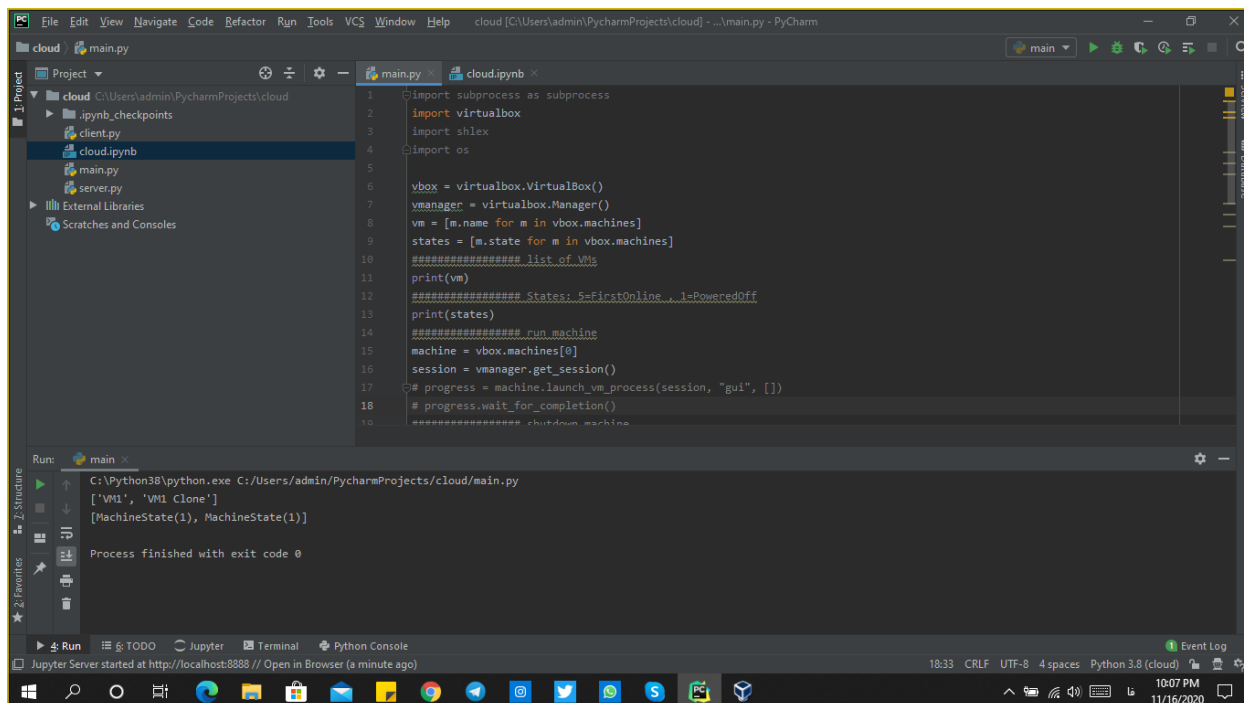
نصب و پیکربندی VM1 طبق صورت تمرین:



نصب و پیکربندی VM2 (VM1 Clone):



لیست ماشین های مجازی موجود در سیستم میزبان و وضعیت هر ماشین مجازی موجود در سیستم:



```
1 import subprocess as subprocess
2 import virtualbox
3 import shlex
4 import os
5
6 vbox = virtualbox.VirtualBox()
7 vmanager = virtualbox.Manager()
8 vm = [m.name for m in vbox.machines]
9 states = [m.state for m in vbox.machines]
10 ##### list of VMs
11 print(vm)
12 ##### States: 5=FirstOnline...1=PoweredOff
13 print(states)
14 ##### run machine
15 machine = vbox.machines[0]
16 session = vmanager.get_session()
17 # progress = machine.launch_vm_process(session, "gui", [])
18 # progress.wait_for_completion()
19 ##### shutdown machine
```

Run: main x

C:\Python38\python.exe C:/Users/admin/PycharmProjects/cloud/main.py

['VM1', 'VM1 Clone']

[MachineState(1), MachineState(1)]

Process finished with exit code 0

وضعیت هر ماشین با یک عدد مشخص می شود که عدد ۱ به معنی خاموش بودن ماشین و عدد ۵ به معنی روشن بودن ماشین است.

روشن و خاموش کردن ماشین مجازی:

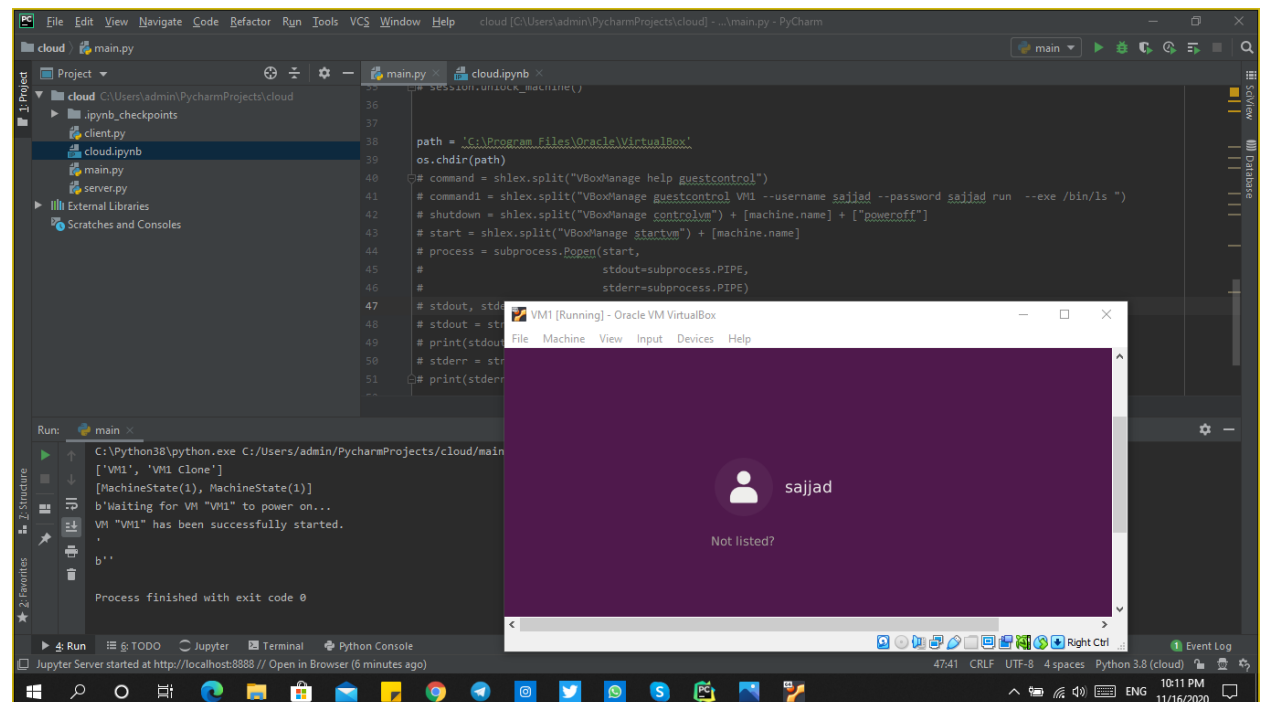
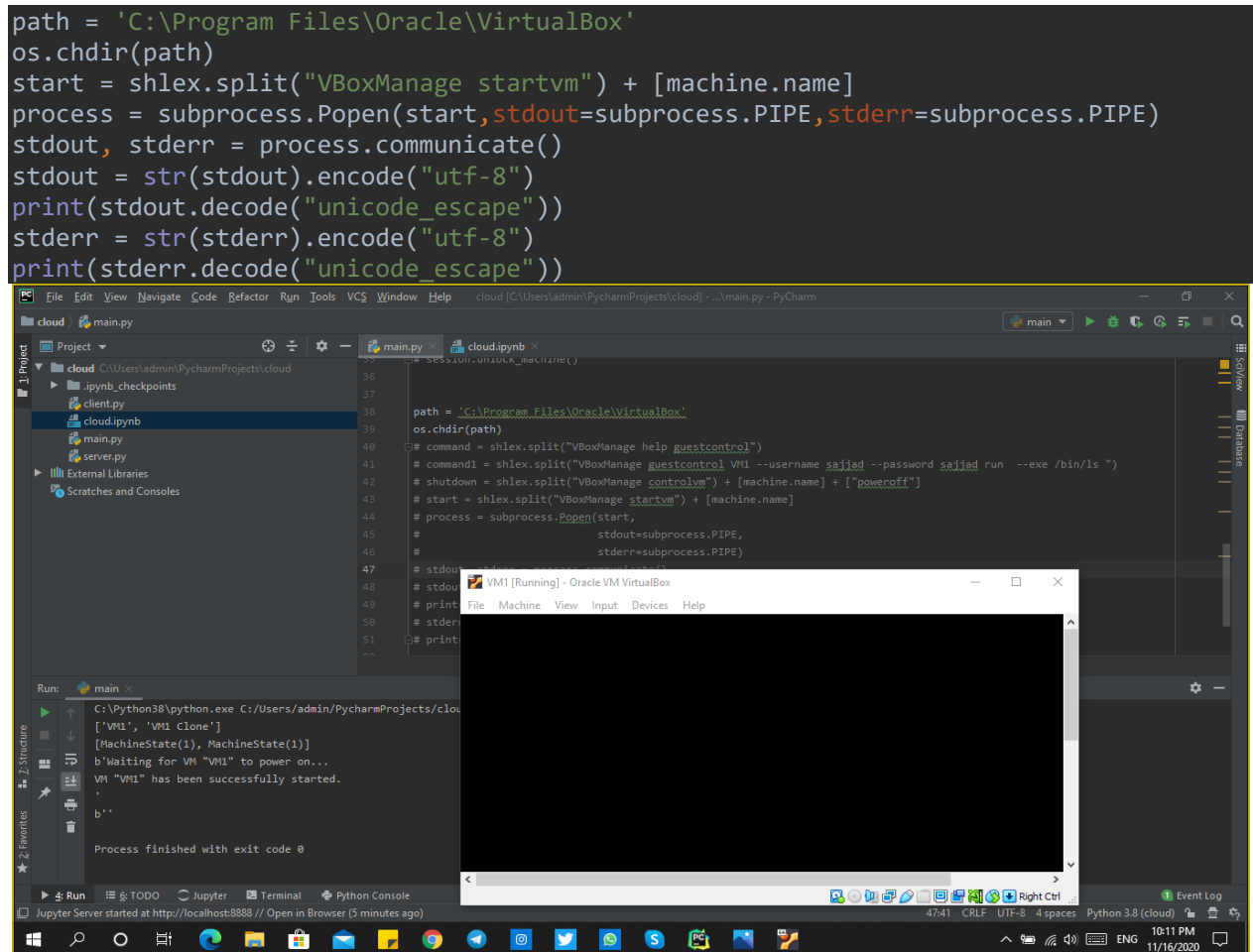
برای روشن کردن ماشین ما دو دستور را امتحان نمودیم:

دستور اول: با استفاده از api موجود در pyvbox:

```
##### run machine
machine = vbox.machines[0]
session = vmanager.get_session()
progress = machine.launch_vm_process(session, "gui", [])
progress.wait_for_completion()
```

دستور دوم: با استفاده از VboxManage خود VirtualBox که از طریق دستور ترمینال cmd در آدرس نصب VirtualBox قابل دسترسی است. این دستور را با پایتون parse نموده و ماشین مجازی را روشن میکنیم (در تصاویر ما از این دستور استفاده کردیم):

```
import subprocess as subprocess
import shlex
import os
##### run machine
```



همچنین ما با استفاده از دستور زیر در یکی از ماشین ها رابط گرافیکی برای آن قرار دادیم:

```
sudo apt install virtualbox-guest-dkms virtualbox-guest-x11 virtualbox-guest-utils
```

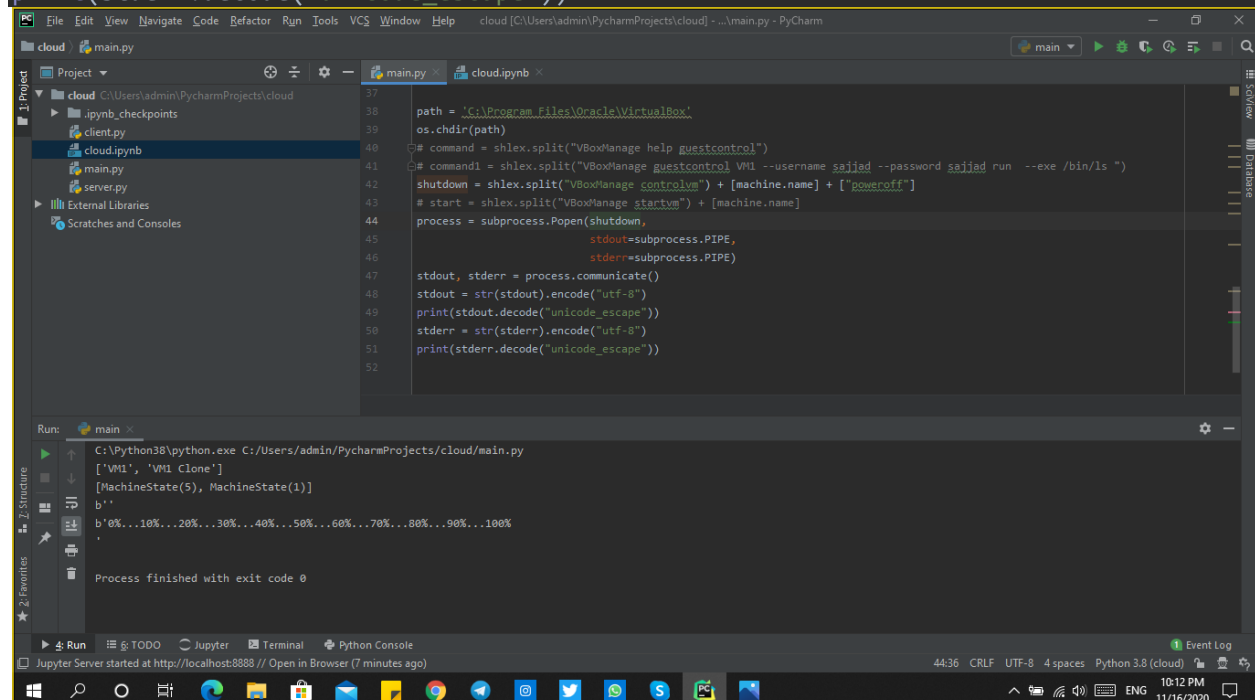
برای خاموش کردن ماشین هم دو دستور امتحان نمودیم:

دستور اول: با استفاده از api موجود در pyvbox:

```
##### shutdown machine
session.console.power_down()
```

دستور دوم: با استفاده از VBoxManage خود VirtualBox (در تصویر ما از این دستور استفاده کردیم):

```
import subprocess as subprocess
import shlex
import os
##### shutdown machine
path = 'C:\Program Files\Oracle\VirtualBox'
os.chdir(path)
shutdown = shlex.split("VBoxManage controlvm") + [machine.name] + ["poweroff"]
process = subprocess.Popen(shutdown, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
stdout, stderr = process.communicate()
stdout = str(stdout).encode("utf-8")
print(stdout.decode("unicode_escape"))
stderr = str(stderr).encode("utf-8")
print(stderr.decode("unicode_escape"))
```

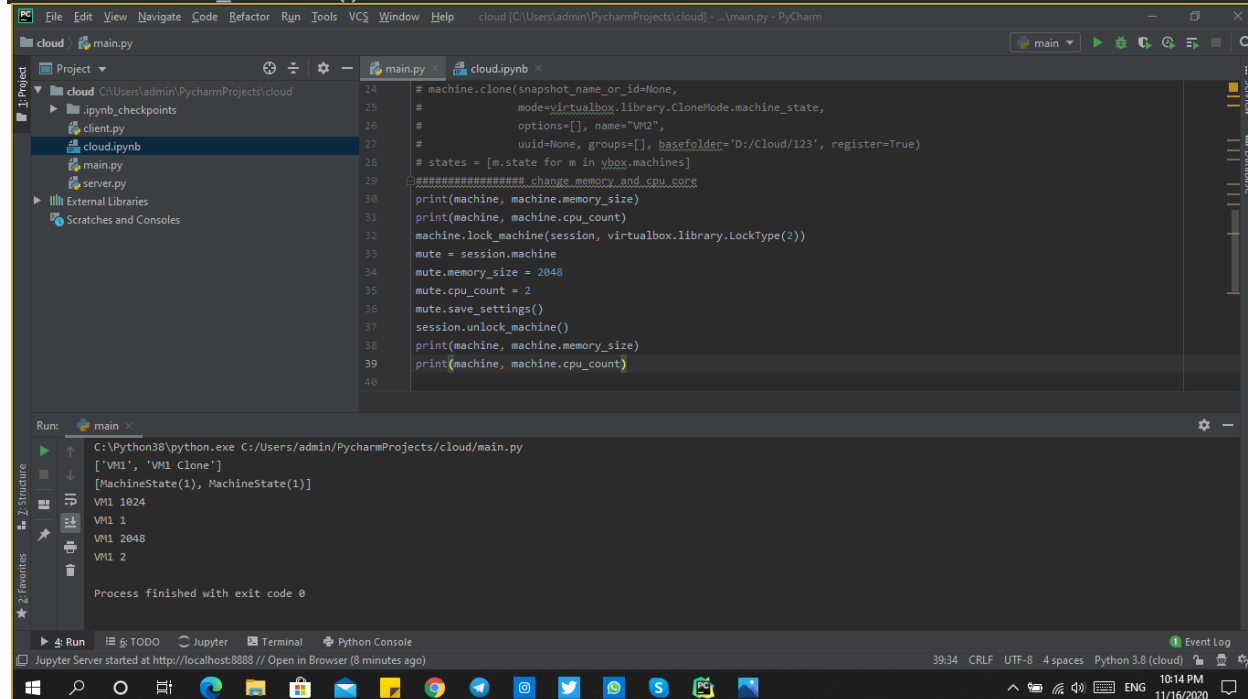


تغییر تعداد هسته های CPU و حافظه اصلی ماشین مجازی:

برای تغییر تعداد هسته و حافظه اصلی ابتدا می بایستی ماشین را lock می کردیم (Locktype آن می باید از نوع write باشد) سپس آن را mute می کردیم و تغییرات لازم روی آن اعمال و ذخیره نموده و سپس ماشین را unlock می کردیم (طبق بخش ۳.۲ منبع رسمی VirtualBox که لینک آن در قسمت منابع آمده است).
کد این قسمت به صورت زیر است:

```
machine = vbox.machines[0]
session = vmanager.get_session()

##### change memory and cpu core
machine.lock_machine(session, virtualbox.library.LockType(2))
mute = session.machine
mute.memory_size = 2048
mute.cpu_count = 2
mute.save_settings()
session.unlock_machine()
```



ایجاد ماشین مجازی (Clone کردن یک ماشین):

برای Clone کردن یک ماشین کد زیر را اجرا کردیم و نام آن را VM2 قرار دادیم (از VM1 عمل Clone را انجام دادیم):

```
machine = vbox.machines[0]
session = vmanager.get_session()
```

```
##### clone machine
machine.clone(snapshot_name_or_id=None,
              mode=virtualbox.library.CloneMode.machine_state,
              options=[], name="VM2",
              uuid=None, groups=[], basefolder='D:/Cloud/123', register=True)
```

```
19 ##### shutdown_machine
20 # session.console.power_down()
21 ##### delete_machine
22 # machine.remove("delete=true")
23 ##### clone_machine
24 print(vm)
25 print(states)
26 machine.clone(snapshot_name_or_id=None,
27              mode=virtualbox.library.CloneMode.machine_state,
28              options=[], name="VM2",
29              uuid=None, groups=[], basefolder='D:/Cloud/123', register=True)
30 print(vm)
31 print(states)
32 # states = [m.state for m in vbox.machines]
33 ##### change memory and cpu cores
34 # print(machine, machine.memory_size)
35 # print(machine, machine.cpu_count)
```

Run: main

C:\Python38\python.exe C:/Users/admin/PycharmProjects/cloud/main.py

['VM1', 'VM1 Clone', 'VM2']

[MachineState(1), MachineState(1), MachineState(1)]

['VM1', 'VM1 Clone', 'VM2']

[MachineState(1), MachineState(1), MachineState(1)]

Process finished with exit code 0

حذف ماشین مجازی: از دستور زیر استفاده میکنیم و VM2 را حذف می کنیم:

```
machine = vbox.machines[2]
machine.remove("delete=true")
```

```
16 session = vmanager.get_session()
17 # progress = machine.launch_vm_process(session, "gui", [])
18 # progress.wait_for_completion()
19 ##### shutdown_machine
20 # session.console.power_down()
21 ##### delete_machine
22 print(vm)
23 print(states)
24 machine.remove("delete=true")
25 ##### clone_machine
26 # machine.clone(snapshot_name_or_id=None,
27 #               mode=virtualbox.library.CloneMode.machine_state,
28 #               options=[], name="VM2",
29 #               uuid=None, groups=[], basefolder='D:/Cloud/123', register=True)
30 print(vm)
31 print(states)
32 # states = [m.state for m in vbox.machines]
```

Run: main

C:\Python38\python.exe C:/Users/admin/PycharmProjects/cloud/main.py

['VM1', 'VM1 Clone']

[MachineState(1), MachineState(1)]

['VM1', 'VM1 Clone']

[MachineState(1), MachineState(1)]

Process finished with exit code 0

اجرای دستور روی ماشین مجازی از طریق رابط کاربری برنامه: برای اجرای دستورات از VboxManage خود VirtualBox که از طریق دستور ترمینال cmd در آدرس نصب VirtualBox نیز قابل دسترسی است استفاده کردیم.

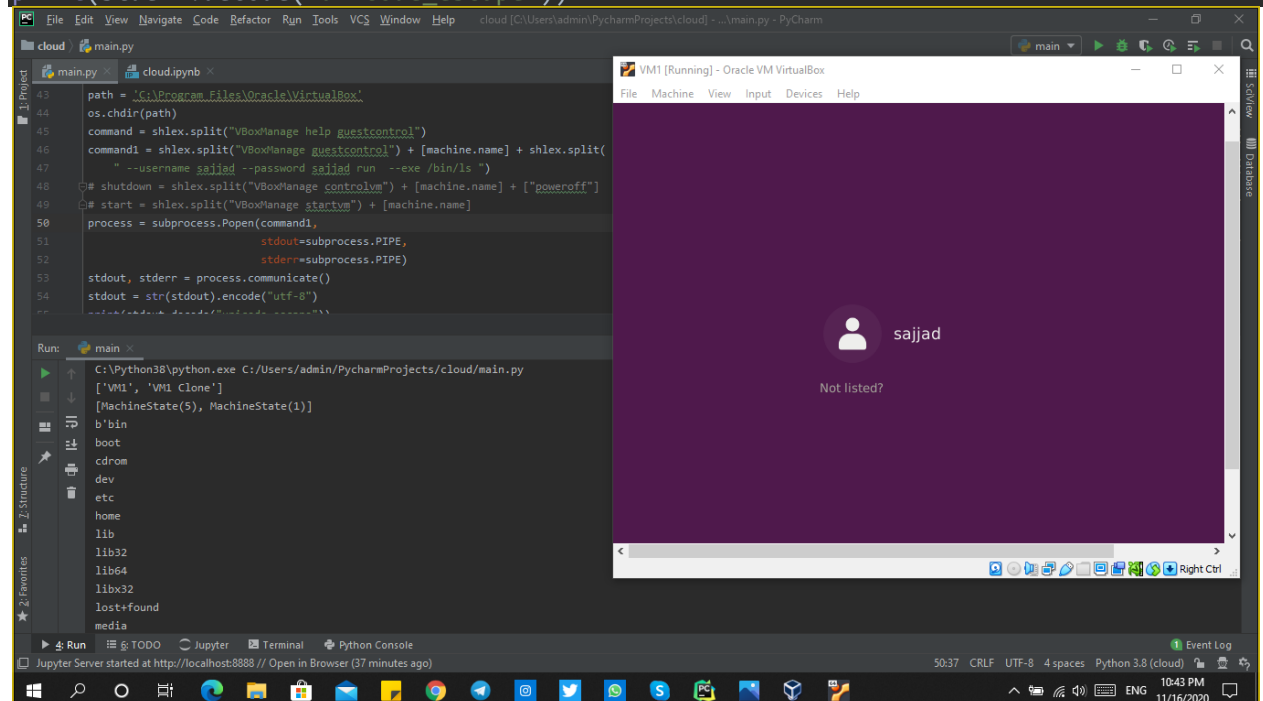
همانطور که در بالا گفته شده بود ما دستورات مربوط به VboxManage را به صورت زیر با پایتون parse می کنیم و خروجی را در ترمینال خود پایتون چاپ می کنیم.

برای این قسمت ما دستور ls را امتحان کردیم و خروجی آن را در ترمینال پایتون چاپ کردیم.

کد این قسمت به صورت زیر است:

```
import subprocess as subprocess
import shlex
import os

path = 'C:\Program Files\Oracle\VirtualBox'
os.chdir(path)
command = shlex.split("VBoxManage help guestcontrol")
command1 = shlex.split("VBoxManage guestcontrol") + [machine.name] + shlex.split(
    "--username sajjad --password sajjad run --exe /bin/ls ")
process = subprocess.Popen(command1, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
stdout, stderr = process.communicate()
stdout = str(stdout).encode("utf-8")
print(stdout.decode("unicode_escape"))
stderr = str(stderr).encode("utf-8")
print(stderr.decode("unicode_escape"))
```

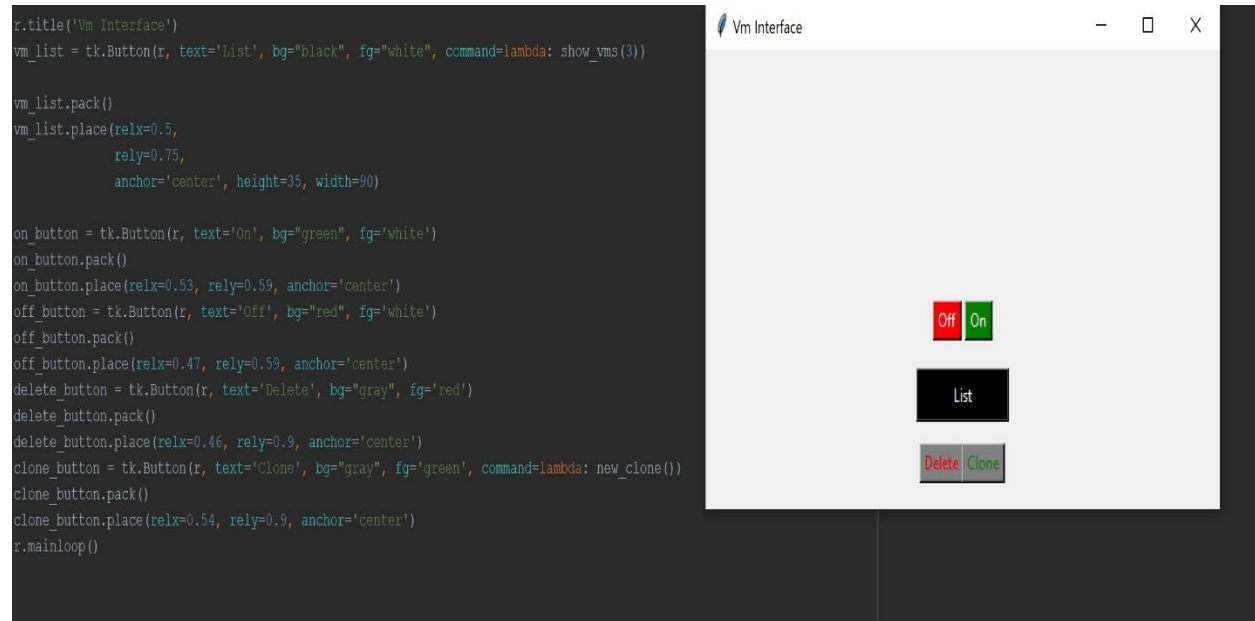


The screenshot shows the PyCharm IDE with a Python script in the main editor. The script uses subprocess to run VBoxManage guestcontrol. The VM window shows a login screen for 'sajjad' with the message 'Not listed?'. The bottom status bar shows the system clock as 10:43 PM on 11/16/2020.

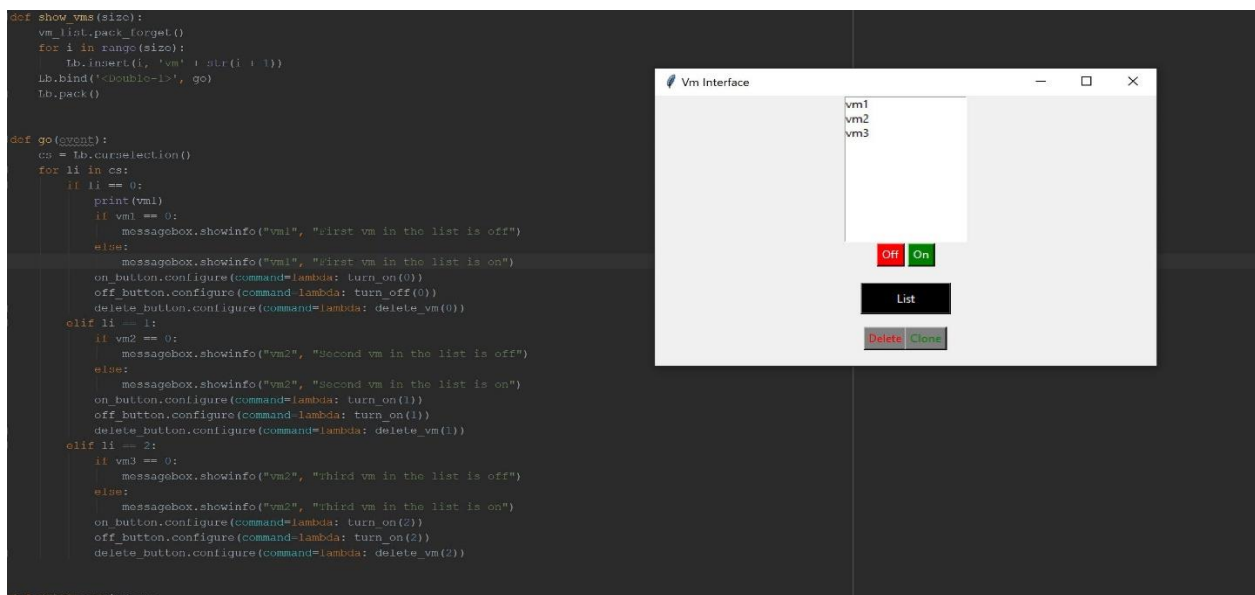
رابط کاربری گرافیکی:

این قسمت با استفاده از GUI tkinter زده شده است و دارای نمایش لیست ماشین‌های مجازی، وضعیت آن‌ها در هر لحظه‌ی درخواستی، کلید روشن و خاموش، کلید پاک کردن یک ماشین و کلید اضافه کردن یک ماشین است.

در تصویر پایین باتن‌های اولیه به همراه کد آن‌ها را مشاهده می‌کنید:

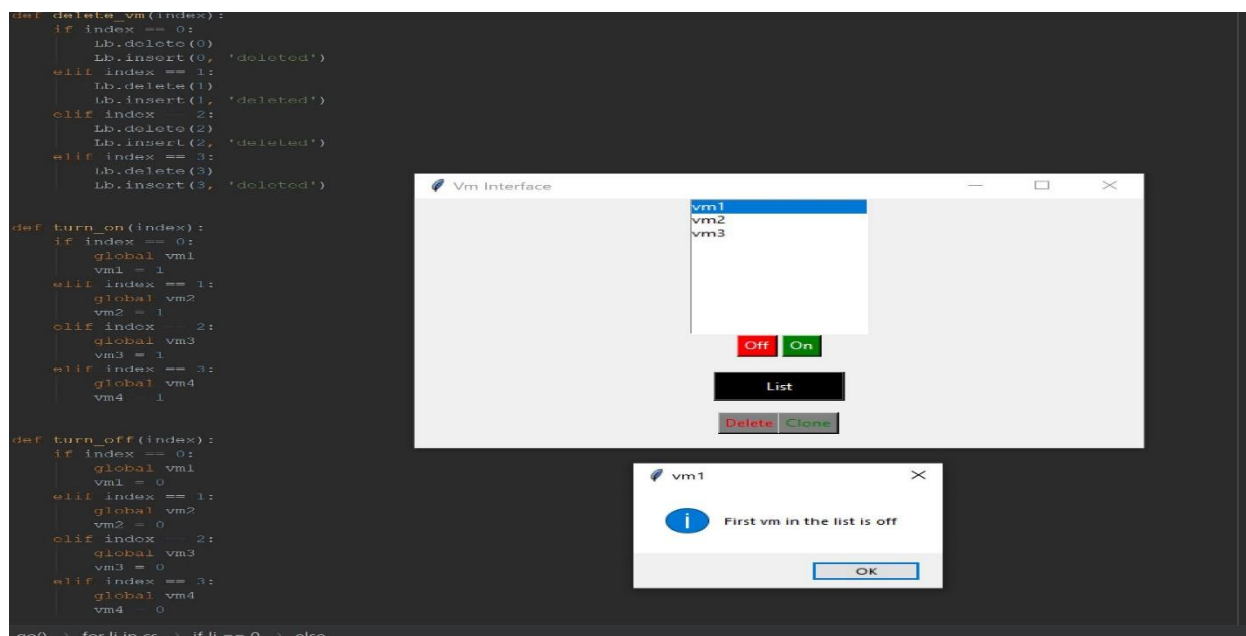


حال با زدن دکمه‌ی list داریم:

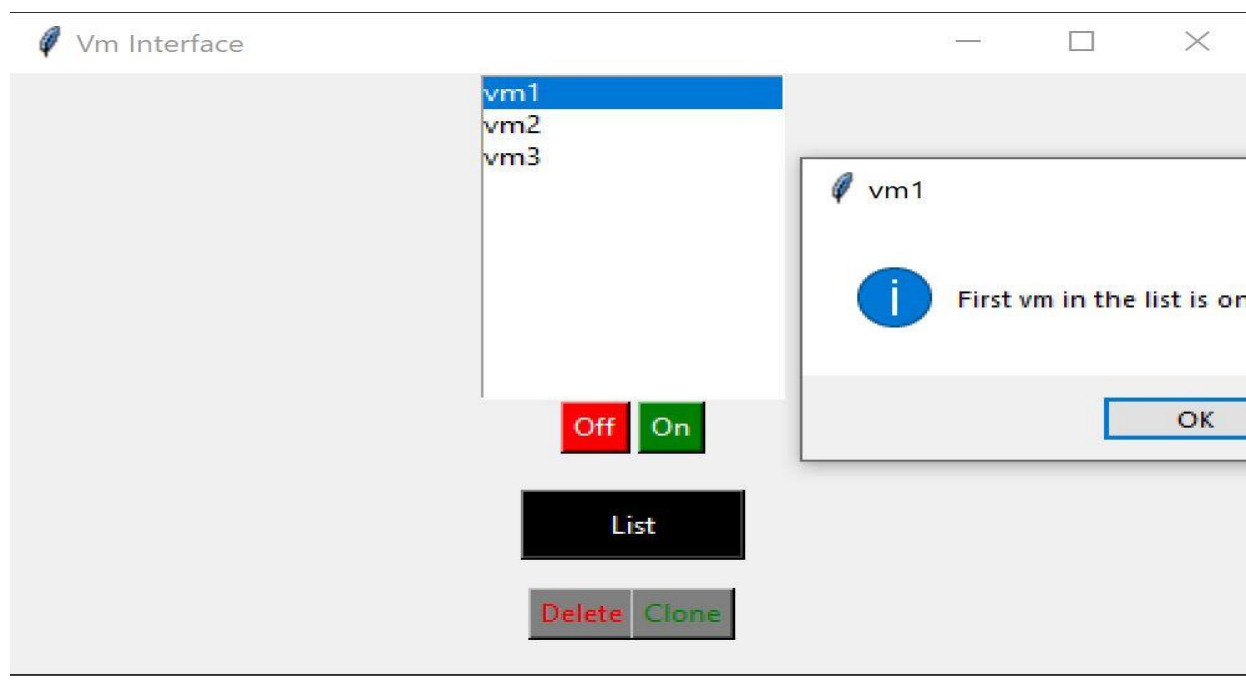


در ادامه به ترتیب تصاویری از کد و نتیجه‌ی گرافیکی آن در رابطه با نشان دادن وضعیت، پاک کردن یک ماشین و اضافه کردن آن را خواهیم دید:

خاموش کردن ماشین



روشن کردن ماشین



حذف ماشین

```
def delete_vm(index):
    if index == 0:
        Lb.delete(0)
        Lb.insert(0, 'deleted')
    elif index == 1:
        Lb.delete(1)
        Lb.insert(1, 'deleted')
    elif index == 2:
        Lb.delete(2)
        Lb.insert(2, 'deleted')
    elif index == 3:
        Lb.delete(3)
        Lb.insert(3, 'deleted')
```

vm1
vm2
deleted

Off On

List

Delete Clone

اضافه کردن ماشین

```
def new_clone():
    global new_clone_n
    Lb.delete(new_clone_n)
    Lb.insert(new_clone_n, 'vm' + str(new_clone_n + 1))
    new_clone_n += 1
    if new_clone_n < 0:
        new_clone_n = 2

r.title('Vm Interface')
vm_list = tk.Button(r, text='List', bg="black", fg="white", command=lambd

vm_list.pack()
vm_list.place(relx=0.5,
              rely=0.75,
              anchor='center', height=35, width=90)

on_button = tk.Button(r, text='On', bg="green", fg='white')
on_button.pack()
```

Vm Interface

vm1
deleted
vm3

Off On

List

Delete Clone

پیکربندی SSH:

تمامی مراحل این بخش با کمک راهنمای قرار گرفته در سایت درس انجام شده است و مراحل آن بدیهی است و تنها به آوردن تصاویر آن بسنده می کنیم:

```
alishf@shf:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/alishf/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alishf/.ssh/id_rsa
Your public key has been saved in /home/alishf/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:3S2GLafz9d3rkVc1WEZtuVPx+UoJ6sgg8mJiaVk5FkE alishf@shf
The key's randomart image is:
+---[RSA 3072]-----+
| .E . . . . .|=+
| . . . . .+.*|
| . . . . .O +. +=|
| . . . . .O * . ++|
| . . . . .S B O + +|
| =O.. O + = O O=|
| .O . . . . .O=|
| oo . . . . .+|
| . . . . .O |
+-----[SHA256]-----+
alishf@shf:~$ ls ~/.ssh/
id_rsa id_rsa.pub known_hosts
alishf@shf:~$ scp ~/.ssh/id_rsa.pub alishf@192.168.1.15:./
alishf@192.168.1.15's password:
id_rsa.pub                                100% 564 335.9KB/s 00:00
alishf@shf:~$ ssh alishf@192.168.1.15
alishf@192.168.1.15's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-52-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Wed 18 Nov 2020 12:23:41 PM UTC

System load: 0.08          Processes:              117
Usage of /: 25.0% of 18.57GB Users logged in:                1
Memory usage: 22%          IPv4 address for enp0s3: 192.168.1.15
Swap usage: 0%

 * Introducing self-healing high availability clustering for MicroK8s!
   Super simple, hardened and opinionated Kubernetes for production.

   https://microk8s.io/high-availability

92 updates can be installed immediately.
15 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Web console: https://shf:9090/ or https://192.168.1.15:9090/
Last login: Wed Nov 18 12:21:19 2020
```

```
alishf@shf:~$ mkdir -p ~/.ssh
alishf@shf:~$ cat id_rsa.pub >> ~/.ssh/authorized_keys
alishf@shf:~$ chmod _R go= ~/.ssh
chmod: invalid mode: '_R'
Try 'chmod --help' for more information.
alishf@shf:~$ chmod -R go= ~/.ssh
alishf@shf:~$ exit
logout
Connection to 192.168.1.15 closed.
```

و در انتها نتیجه‌ی نهایی به درستی به دست می‌آید:

```
alishf@shf:~$ ssh alishf@192.168.1.15 ls
getdeb-repository_0.1-1~getdeb1_all.deb
getdeb-repository_0.1-1~getdeb1_all.deb.1
getdeb-repository_0.1-1~getdeb1_all.deb.2
id_rsa.pub
alishf@shf:~$ ssh alishf@192.168.1.15 mkdir tmp
alishf@shf:~$ ssh alishf@192.168.1.15 ls
getdeb-repository_0.1-1~getdeb1_all.deb
getdeb-repository_0.1-1~getdeb1_all.deb.1
getdeb-repository_0.1-1~getdeb1_all.deb.2
id_rsa.pub
tmp
```


- [1]<https://reviews.financesonline.com/p/oracle-vm-virtualbox>
- [2]<https://reviews.financesonline.com/p/xen-project>
- [3]<https://www.trustradius.com/compare-products/citrix-hypervisor-vs-oracle-vm-virtualbox>
- [4]<https://pydoc.net/pyvbox/1.0.0>
- [5]<https://buildmedia.readthedocs.org/media/pdf/pyvbox/latest/pyvbox.pdf>
- [6]<https://download.virtualbox.org/virtualbox/SDKRef.pdf>
- [7]<https://github.com/sethmlarson/virtualbox-python>
- [8]<https://sysplay.in/blog/linux/2019/07/running-commands-on-virtualbox-from-outside>