

بخش اول

(الف)

خیلی خلاصه بخوایم بگیم، مجازی‌سازی می‌شه استفاده بهینه از سخت‌افزار به صورتی که به کمک آن می‌توانیم نمونه‌هایی مجازی از زیرساخت یا سخت‌افزار یا حافظه یا واحدهای ذخیره‌سازی و... ایجاد کنیم. مجازی‌سازی توانسته ارتباط میان سخت‌افزار و نرم‌افزار را دگرگون کرده و به عنوان یکی از مبانی اصلی رایانش ابری شناخته می‌شود که به استفاده بهینه و تمام و کمال از ظرفیت‌های رایانش ابری کمک می‌کند.

(ب)

معماری چند اجاره‌ای در رایانش ابری به این معناست که در آن مشتریان مختلف از منابع یکسانی استفاده می‌کنند به علاوه اطلاعات آن‌ها از یک‌دیگر ایزوله است و مشتریان به اطلاعات بقیه دسترسی‌ای ندارند.

مشکلاتی که این معماری دارد عبارتند از:

امنیت: درسته که کاربران اطلاعات هم‌دیگر را نمی‌بینند اما دارند از منابع مشترکی استفاده می‌کنند که می‌تواند امنیت را به خطر بیندازد.

قدرت: هرچقدر هم که این قابلیت به خوبی پیاده شده باشد نمی‌تواند به قدرت حالتی که به صورت اختصاصی شما به منابع دسترسی دارید، باشد.

مدیریت و دسترسی: در این حالت شما دسترسی کامل و تمام را بر منابع ندارید و کمی محدود شده‌اید.

(ج)

در مجازی‌سازی سخت‌افزاری از hypervisor استفاده می‌شود که یک لایه میان سخت‌افزار و نرم‌افزار است و منابع فیزیکی را مدیریت می‌کند و ساختارهایی مجازی مانند پردازنده‌های مجازی را به وجود می‌آورد که این مجازی‌سازی سخت‌افزاری نیز انواعی دارد که عبارتند از:

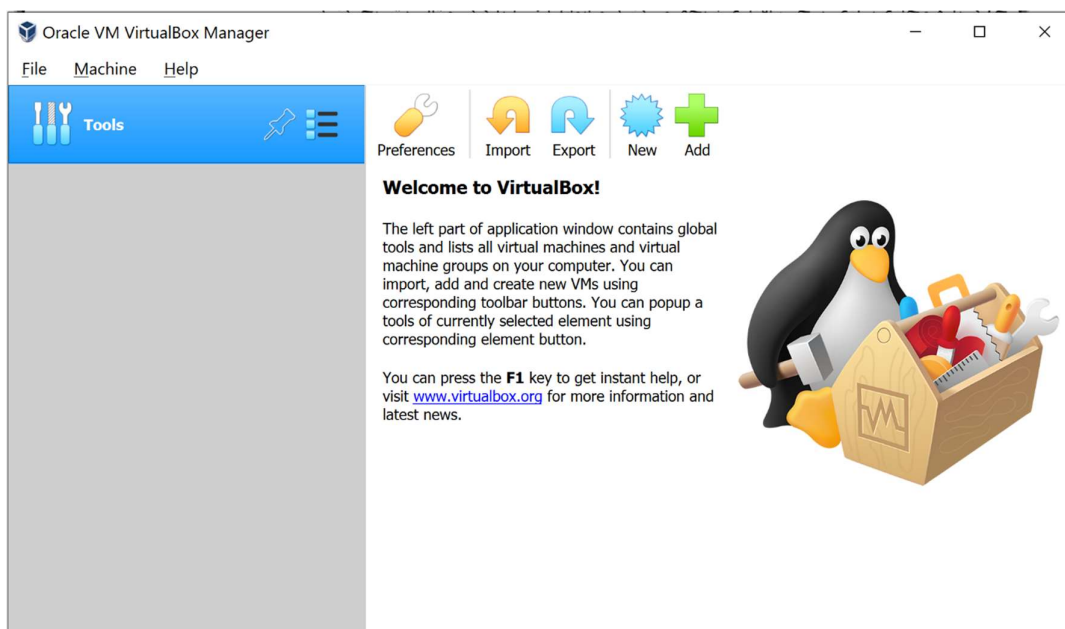
مجازی‌سازی کامل: در این حالت هیچ تغییری برای اجرای برنامه‌ها لازم نیست انجام شود و معماری سخت‌افزاری به طور کامل شبیه‌سازی شده است.

مجازی‌سازی شبیه‌سازی: در این حالت سخت‌افزار توسط ماشین مجازی شبیه‌سازی می‌شود.

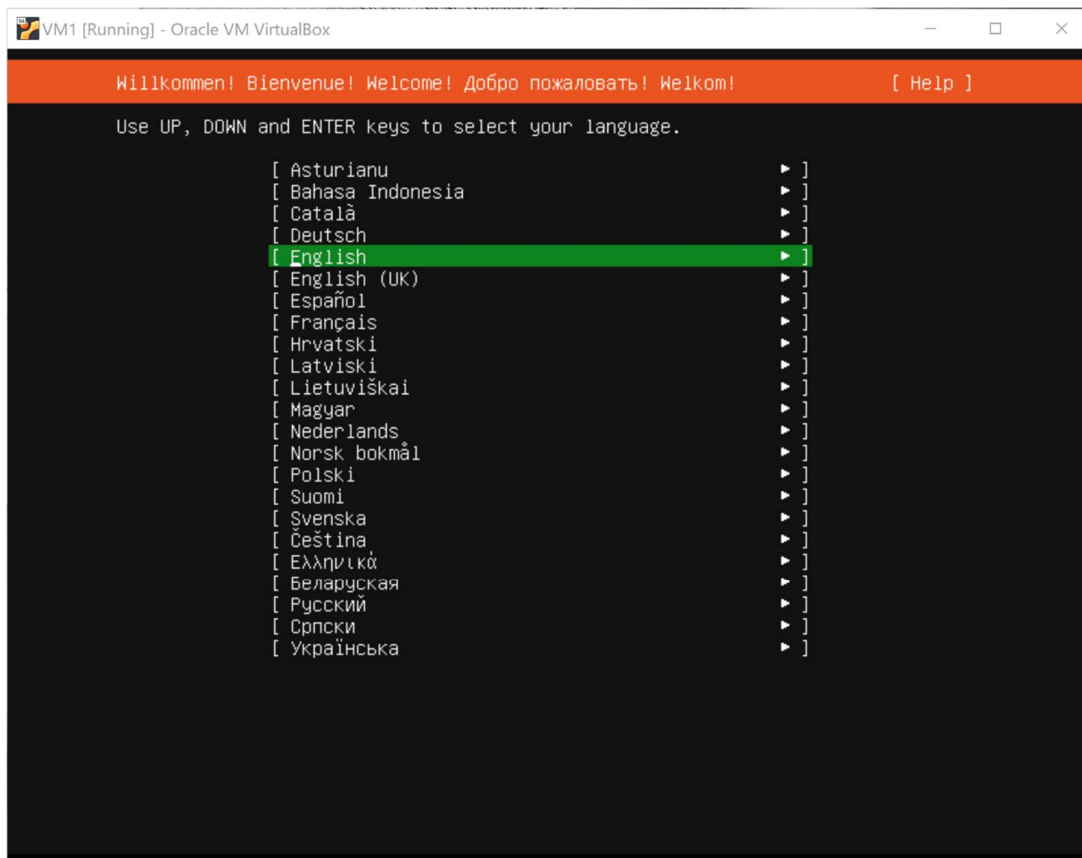
نیمه مجازی‌سازی: در این حالت سخت‌افزار اصلاً شبیه‌سازی نمی‌شود.

بخش دوم

ابتدا نرم افزار virtualbox را نصب می کنیم.



سپس با توجه به اسلایدهای مربوط به ایجاد ماشین مجازی، شروع به ایجاد ماشین مجازی VM1 می کنیم:



در نهایت این ماشین آماده می‌شود:

با قرار دادن شبکه‌ی این ماشین بر روی bridge یک ip به این ماشین اختصاص می‌یابد:

```

ali@vm1:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.40 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::a00:27ff:fe7a:9ed9 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:7a:9e:d9 txqueuelen 1000 (Ethernet)
    RX packets 854 bytes 1075550 (1.0 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 654 bytes 52194 (52.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 212 bytes 17038 (17.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 212 bytes 17038 (17.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ali@vm1:~$
    
```

همان‌طور که می‌بینیم ماشین مجازی با کامپیوتر میزبان در یک سابنت هستند:

```

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix  . : 
IPv4 Address. . . . . : 192.168.1.37
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.1.1
    
```

می‌توانیم به این ماشین از طریق ssh دسترسی داشته باشیم:

```

PS C:\Users\Ali> ssh 192.168.1.40
The authenticity of host '192.168.1.40 (192.168.1.40)' can't be established.
ECDSA key fingerprint is SHA256:hbiPABdg0x7yEks+4eoz3P1mMyF8oJpi0YG3YMMUXqc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.40' (ECDSA) to the list of known hosts.
ali@192.168.1.40's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-70-generic x86_64)

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

System information as of Thu 01 Apr 2021 12:01:21 PM UTC

System load:  0.01          Processes:      109
Usage of /:   22.1% of 18.57GB Users logged in: 1
Memory usage: 10%          IPv4 address for enp0s3: 192.168.1.40
Swap usage:   0%

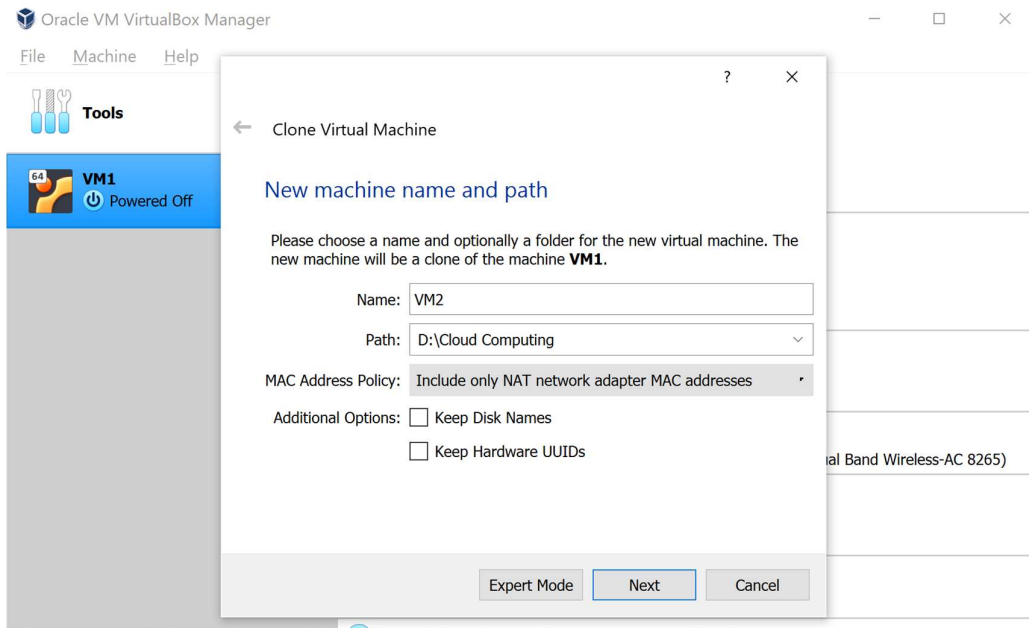
 * Introducing self-healing high availability clusters in MicroK8s.
   Simple, hardened, Kubernetes for production, from RaspberryPi to DC.

   https://microk8s.io/high-availability

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

Last login: Thu Apr  1 10:34:29 2021
ali@vm1:~$
    
```

سپس ماشین VM2 را از طریق clone کردن ماشین اولی ایجاد می‌کنیم:



با قرار دادن شبکه‌ی این ماشین نیز بر روی bridge برای آن یک ip می‌گیریم:

```
sudo ip address flush scope global dynamic
```

```
sudo dhclient -v
```

```

VM2 [Running] - Oracle VM VirtualBox
ali@vm1:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.41 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::a00:27ff:fe42:d9b prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:42:0d:9b txqueuelen 1000 (Ethernet)
    RX packets 82 bytes 12805 (12.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 71 bytes 8938 (8.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 94 bytes 7340 (7.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 94 bytes 7340 (7.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ali@vm1:~$ _
    
```

و از طریق ssh می‌توانیم به این ماشین متصل شویم:

```
PS C:\Users\Ali> ssh 192.168.1.41
The authenticity of host '192.168.1.41 (192.168.1.41)' can't be established.
ECDSA key fingerprint is SHA256:hb1PABdg0x7yEks+4eoz3P1mMyF8oJpi0YG3YMMUXqc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.41' (ECDSA) to the list of known hosts.
ali@192.168.1.41's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-70-generic x86_64)

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

System information as of Thu 01 Apr 2021 12:38:26 PM UTC

System load:  0.0          Processes:      109
Usage of /:   22.1% of 18.57GB   Users logged in: 1
Memory usage: 9%          IPv4 address for enp0s3: 192.168.1.41
Swap usage:  0%

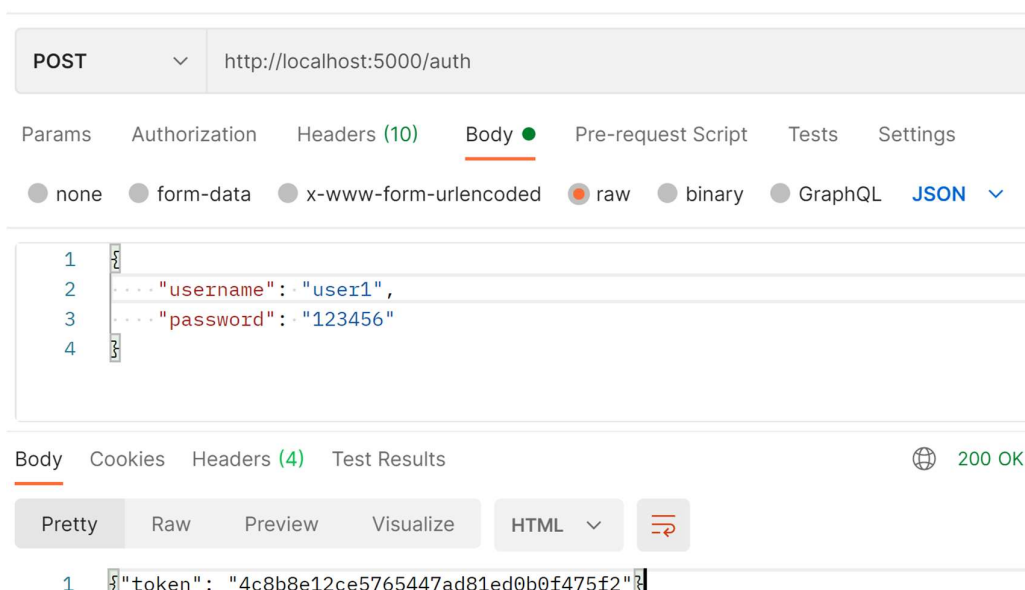
 * Introducing self-healing high availability clusters in MicroK8s.
   Simple, hardened, Kubernetes for production, from RaspberryPi to DC.

   https://microk8s.io/high-availability

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

Last login: Thu Apr  1 12:30:06 2021
ali@vm1:~$
```

حال از طریق سرور پایتونی‌ای که فراهم کردیم به سراغ مدیریت سرورها می‌رویم. این سرور به کمک زبان پایتون و کتابخانه‌ی flask فراهم شده است و برای پیاده‌سازی ارتباطات لازم با ماشین‌ها هم در تعدادی از نیازمندی‌ها از یکی از کتابخانه‌های ارتباطی پایتون و virtualbox استفاده شده و مواردی که در این کتابخانه موجود نبودند نیز با کمک دستورات خود virtualbox یعنی vboxmanage انجام شده است. در ابتدا به صورت زیر باید در سیستم به عنوان یوزر admin یا user1 لاگین کنیم:



در قسمت اول می‌توانیم وضعیت تمامی ماشین‌ها را ببینیم:

GET
http://localhost:5000

Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings

none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON

```

1 {
2   ... "command": "status"
3 }

```

Body
Cookies
Headers (4)
Test Results
200 OK 1436 ms


Pretty
Raw
Preview
Visualize
JSON


```


1 {
2   "command": "status",
3   "details": [
4     {
5       "vmName": "VM1",
6       "status": "off"
7     },
8     {
9       "vmName": "VM2",
10      "status": "off"
11     }
12   ]
13 }


```


در این قسمت می‌توانیم وضعیت یک ماشین خاص را ببینیم:

HW1_CC / status VM 

Save 

GET  http://localhost:5000


Params Authorization Headers (10) **Body**  Pre-request Script Tests Settings



☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** 

```

1  {
2    "command": "status",
3    "vmName": "VM1"
4  }

```

Body Cookies Headers (4) Test Results  200 OK 878 ms 2


Pretty Raw Preview Visualize **JSON**  


```


1  {
2    "command": "status",
3    "vmName": "VM1",
4    "status": "on"
5  }

```

در این قسمت می‌توانیم یک ماشین را خاموش یا روشن کنیم:

GET  http://localhost:5000


Params Authorization Headers (10) **Body**  Pre-request Script Tests Settings



☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** 

```

1  {
2    "command": "on/off",
3    "vmName": "VM3"
4  }

```

Body Cookies Headers (4) Test Results  200 OK 618 ms

Pretty Raw Preview Visualize **JSON**  

```

1  {
2    "command": "on/off",
3    "vmName": "VM3",
4    "status": "powering off"
5  }

```

در این قسمت می‌توانیم یک ماشین جدید را از روی یک ماشین قدیمی ایجاد کنیم:

The screenshot shows a REST client interface. At the top, a GET request is configured to `http://localhost:5000`. The 'Body' tab is selected, and the request body is a JSON object: `{ "command": "clone", "sourceVmName": "VM1", "destVmName": "VM3" }`. Below the request, the 'Body' tab of the response is selected, showing a 200 OK status and a JSON response: `{ "command": "clone", "sourceVmName": "VM1", "destVmName": "VM3", "status": "ok" }`.

در این قسمت می‌توان مشخصات و تنظیمات این ماشین مجازی را تغییر داد:

GET http://localhost:5000

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** ▾

```

1 {
2   "command": "setting",
3   "vmName": "VM3",
4   "cpu": 2,
5   "ram": 1024
6 }

```

Body Cookies Headers (4) Test Results 200 OK 668 ms

Pretty Raw Preview Visualize **JSON** ▾

```

1 {
2   "command": "setting",
3   "vmName": "VM3",
4   "cpu": 2,
5   "ram": 1024,
6   "status": "ok"
7 }

```

<div> <div>64</div> <div>VM1</div> <div>Powered Off</div> </div> <div> <div>64</div> <div>VM2</div> <div>Powered Off</div> </div> <div> <div>64</div> <div>VM3</div> <div>Powered Off</div> </div>	<div> <div>General</div> <div>Name: VM3</div> <div>Operating System: Ubuntu (64-bit)</div> </div> <div> <div>System</div> <div>Base Memory: 1024 MB</div> <div>Processors: 2</div> <div>Boot Order: Floppy, Optical, Hard Disk</div> <div>Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization</div> </div>
--	--

در این قسمت می‌توان یک ماشین را حذف کرد:

GET
http://localhost:5000

Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings

none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON

```

1 {
2   ... "command": "delete",
3   ... "vmName": "VM3"
4 }

```

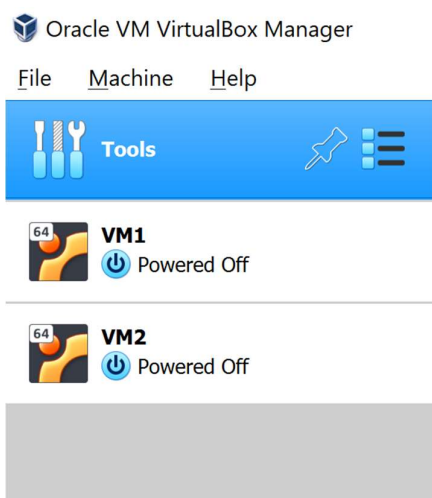
Body
Cookies
Headers (4)
Test Results
200 OK

Pretty
Raw
Preview
Visualize
JSON

```

1 {
2   "command": "delete",
3   "vmName": "VM3",
4   "status": "ok"
5 }

```



در این قسمت می‌خواهیم یک دستور را بر روی ماشین مجازی اجرا کنیم که برای آن ابتدا باید guest addition را بر روی ماشین فعال کنیم که با کمک لینک زیر این کار را انجام دادیم:

<https://linuxize.com/post/how-to-install-virtualbox-guest-additions-in-ubuntu/>

GET http://localhost:5000

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```

1 {
2   "command": "execute",
3   "vmName": "VM1",
4   "input": "ls ~/.ssh"
5 }

```

Body Cookies Headers (4) Test Results 200 OK 1224

Pretty Raw Preview Visualize **JSON**

```

1 {
2   "command": "execute",
3   "vmName": "VM1",
4   "input": "ls ~/.ssh",
5   "response": "authorized_keys\nid_rsa\nid_rsa.pub\nknown_hosts.save\n"
6 }

```

در این قسمت می‌خواهیم یک فایل را بر روی یکی از ماشین‌ها آپلود کنیم:

GET http://localhost:5000

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```

1 {
2   "command": "upload",
3   "vmName": "VM1",
4   "file": "D:/Cloud Computing/server/user.py",
5   "target_path": "/home/ali"
6 }

```

Body Cookies Headers (4) Test Results 200 OK 113

Pretty Raw Preview Visualize **JSON**

```

1 {
2   "command": "upload",
3   "vmName": "VM1",
4   "file": "D:/Cloud Computing/server/user.py",
5   "target_path": "/home/ali",
6   "response": "ok"
7 }

```

و در قسمت پایانی نیز بین ماشین‌ها فایلی را منتقل کردیم:

GET
http://localhost:5000

Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings

none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON

```

2      ...."command": "transfer",
3      ...."originVM": "VM1",
4      ...."originPath": "/home/ali/test.txt",
5      ...."destVM": "VM2",
6      ...."destPath": "/home/ali/"
7      }

```

Body
Cookies
Headers (4)
Test Results
200 OK 782 ms 2

Pretty
Raw
Preview
Visualize
JSON

```

1      {
2        "command": "transfer",
3        "originVM": "VM1",
4        "originPath": "/home/ali/test.txt",
5        "destVM": "VM2",
6        "destPath": "/home/ali/",
7        "response": "ok"
8      }

```