

## Lab. 1 Masina Virtuala - VirtualBox

Ce este o Masina Virtuala si la ce foloseste?

Exemplu: Avem un calculator cu Windows 7 și vrem să folosim un anumit program, dar acesta nu este compatibil cu Windows 7 (programul se blochează și nu poate fi folosit). Ce putem face? Avem mai multe posibilități, fie instalăm Windows xp în loc de 7, fie instalăm Windows xp și apoi 7 într-o configurație dual-boot sau, varianta cea mai bună, instalăm un program de tip mașină virtuală în care instalăm apoi Windows xp.

Prin urmare, o mașină virtuală e un program instalat pe calculator, care ne oferă un calculator virtual în care putem instala un sistem de operare (windows și linux de obicei).

Pe scurt: avem calculatorul nostru pe care instalăm programul de mașină virtuală și pe aceasta mașină instalăm un alt sistem de operare (ex. windows, linux, ubuntu).

### Definiții

#### 1.Mașină virtuală

**Mașina virtuală** este o **aplicație software** care simulează în totalitate funcționarea componentelor **hardware** din componența unui **sistem informatic** (computer).

#### 2.Virtualizare

**Virtualizarea** unui **proces informatic** presupune separarea arhitecturii logice, de configurația suportului fizic prin care acesta este realizat. Prin **virtualizare**, elementele din logica unui **proces informatic** nu sesizează componentele fizice ce fac posibilă execuția acestuia. Prin **virtualizare**, execuția unei aplicații software nu va depinde de configurația hardware.

Tipuri de **virtualizare**: **virtualizarea** accesului, **virtualizarea** aplicațiilor, **virtualizarea** proceselor, **virtualizarea** rețelei de date, **virtualizarea** spațiului de stocare.

### Avantaje

**Virtualizarea** permite proiectarea și exploatarea unor sisteme cu o productivitate ridicată, adaptabile la cerințele operaționale de moment, cu fiabilitate și disponibilitate crescute.

**Virtualizarea** permite reluarea rapidă a activității în caz de dezastru, prin utilizarea unor servere redundante, servere aflate permanent online în altă locație decât cea de bază.

**Virtualizarea** reduce costurile de mentenanță și administrare. Utilizarea imaginilor virtuale permite mutarea serverelor de pe o platformă hardware pe o alta, timpii morți fiind de ordinul minutelor. Prin **virtualizarea** accesului sau al aplicațiilor, deplasarea personalului IT în locația utilizatorului nu mai este necesară.

**Virtualizarea** crește semnificativ nivelul de securitate al informațiilor. În situația virtualizării accesului sau al aplicațiilor, datele nu mai sunt stocate și procesate local la utilizator (nici chiar măcar în memoria RAM). Terminalul utilizatorului va afișa numai informații stocate și procesate în locația server-ului.

**Virtualizarea** eficientizează utilizarea resurselor de calcul, în sensul că, soluția 1 sistem de operare = 1 sistem informatic folosește aproximativ **40%** din resursele de calcul ale sistemului informatic. Prin **virtualizare**, resursele de calcul vor fi folosite în proporție de 100 %.

Etc...etc...etc...

Ce putem face cu o mașină virtuală?

1. Avem programe mai vechi care nu rulează pe versiuni mai noi de windows sau nu rulează pe versiuni de windows pe 64 de biți
2. Vrem să testăm un sistem de operare (de obicei linux) fără a ne complica existența instalându-l direct pe calculator (ceea ce ar putea “corupe” calculatorul)
3. Avem un fișier executabil și nu suntem siguri de proveniența lui. Dacă îl rulăm direct pe calculator e posibil să-l infectăm. Dacă executăm programul în mașina virtuală virusul va fi “încapsulat” în aceasta, evitând astfel infectarea calculatorului (sau a sistemului de operare).

Există mai multe programe de tip mașină virtuală. Pentru laboratoarele de Securitate Informației recomand programul **VirtualBox**.

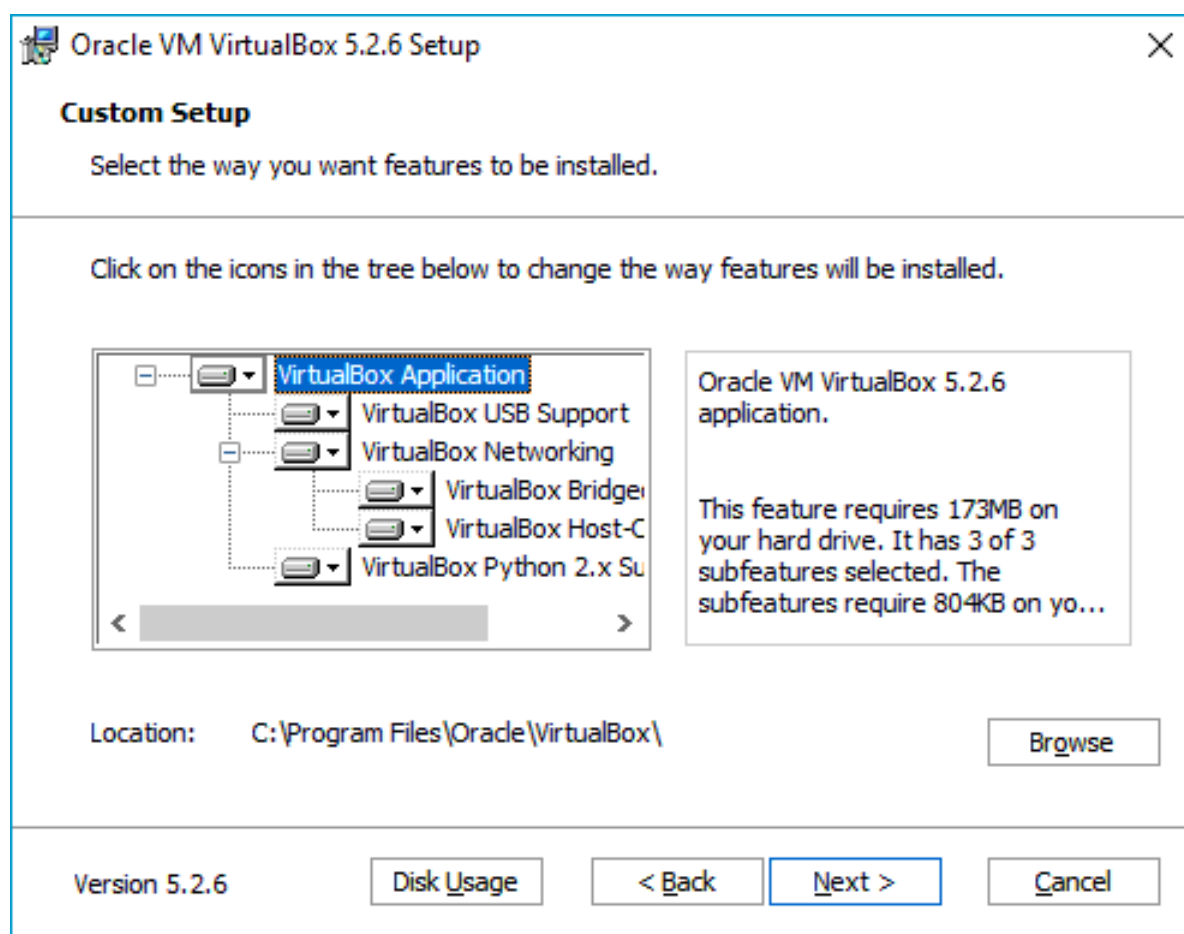
Asadar, VirtualBox este un produs software de virtualizare. Poate fi folosit pe Linux, Solaris, Mac OS X și [Microsoft Windows](https://www.microsoft.com/windows/virtualization/). VirtualBox se află în repositoryele standard Debian, Ubuntu 18.04, 16.04 și Ubuntu 14.04, astfel încât să îl puteți instala cu comanda:

**sudo apt install virtualbox**

VirtualBox poate fi descărcat gratuit de pe site-ul oficial

<https://www.virtualbox.org/wiki/Downloads>

unde sunt oferite versiuni pentru Windows, Mac OS X și Linux.



În timpul instalării VirtualBox, dacă veți lăsa activată componenta de acces la Internet, veți vedea avertismentul «Warning: Network Interfaces», care informează că în procesul de

instalare conexiunea la Internet va fi temporar intrerupta (si se va conecta automat dupa instalarea driverelor si setarea conexiunii).

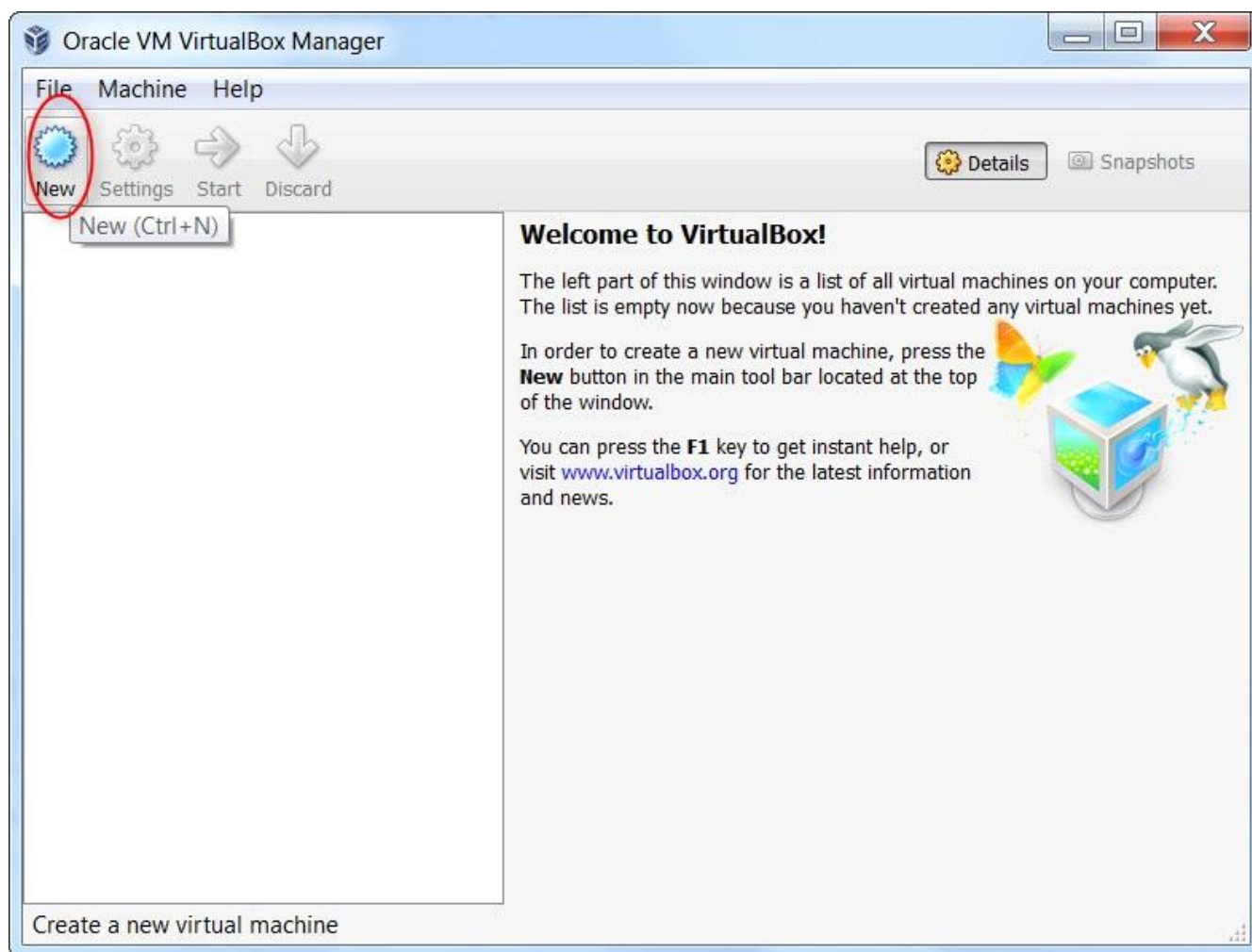
Dupa finalizarea instalarii puteti porni Oracle VM VirtualBox.

### Crearea unei masini virtuale in VirtualBox

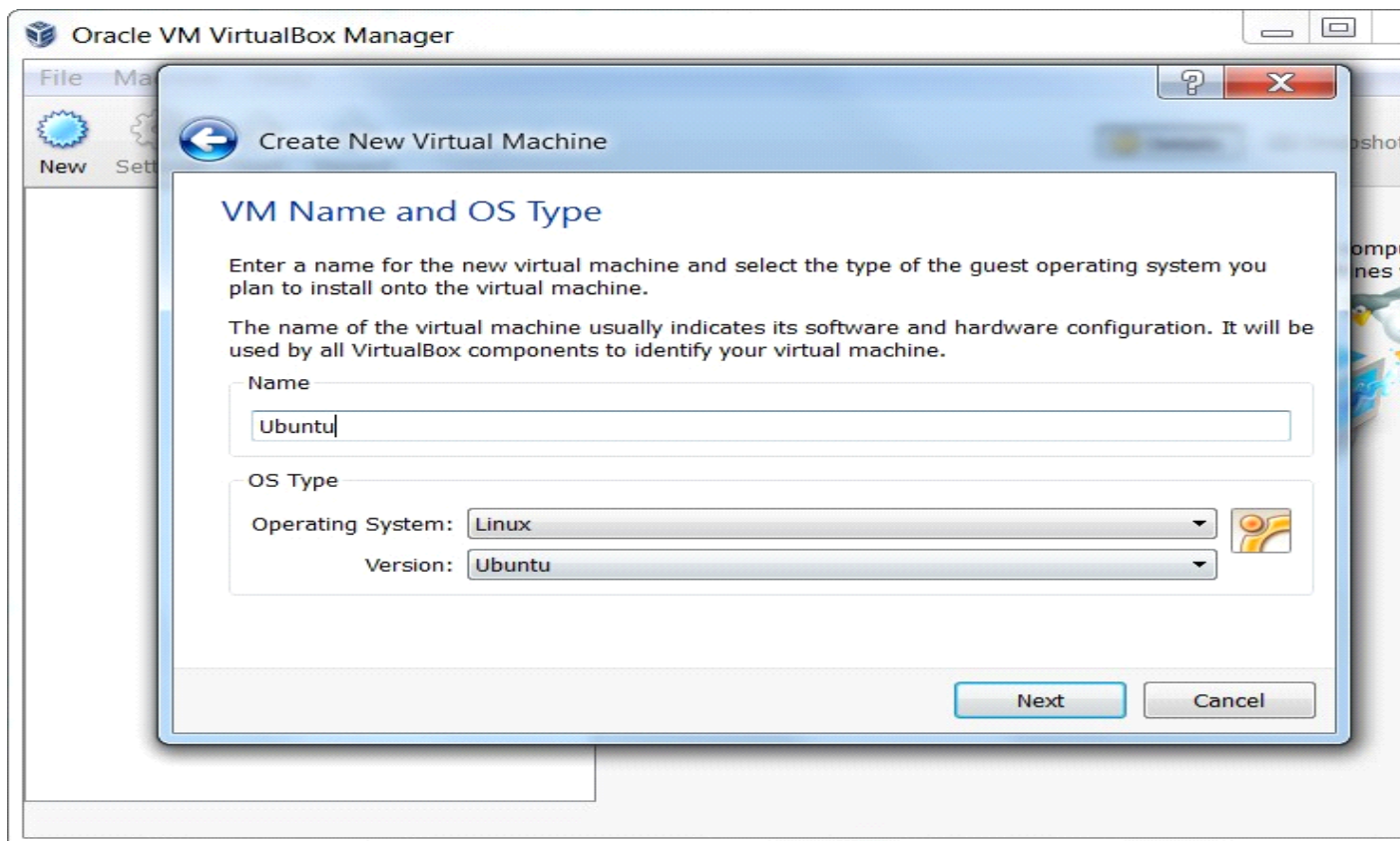
Nota: pentru functionarea masinilor virtuale este necesar, ca pe calculator sa fie activata virtualizarea VT-x sau AMD-V in BIOS. De obicei este activata in mod implicit, dar, daca ceva nu va merge cum trebuie, tineti cont si de acest fapt.

Acum, sa realizam prima noastra masina virtuala. In exemplul de mai jos este utilizat VirtualBox, rulat in Windows, ca un sistem de operare oaspete.

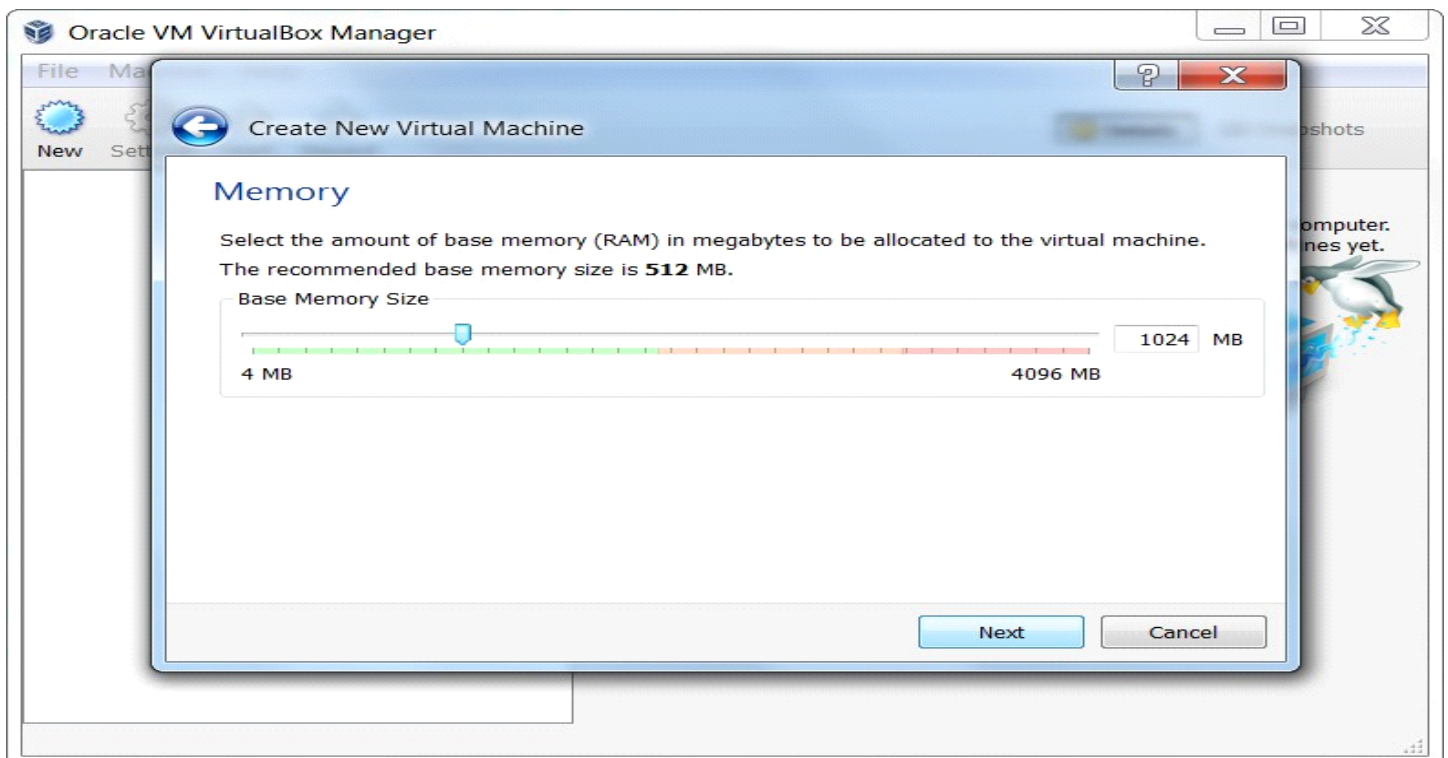
- Apasati «New» in fereastra Oracle VM VirtualBox Manager.



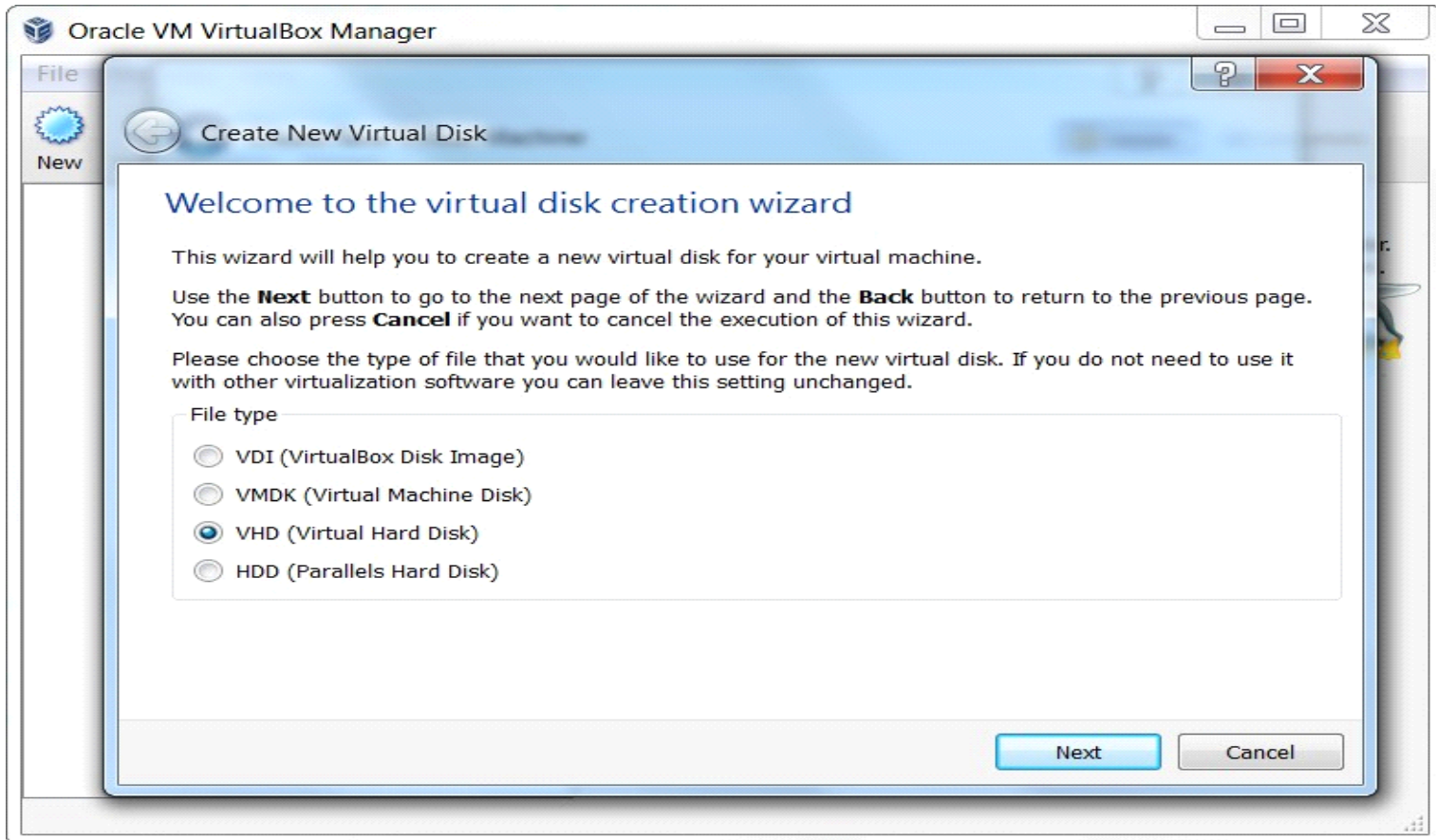
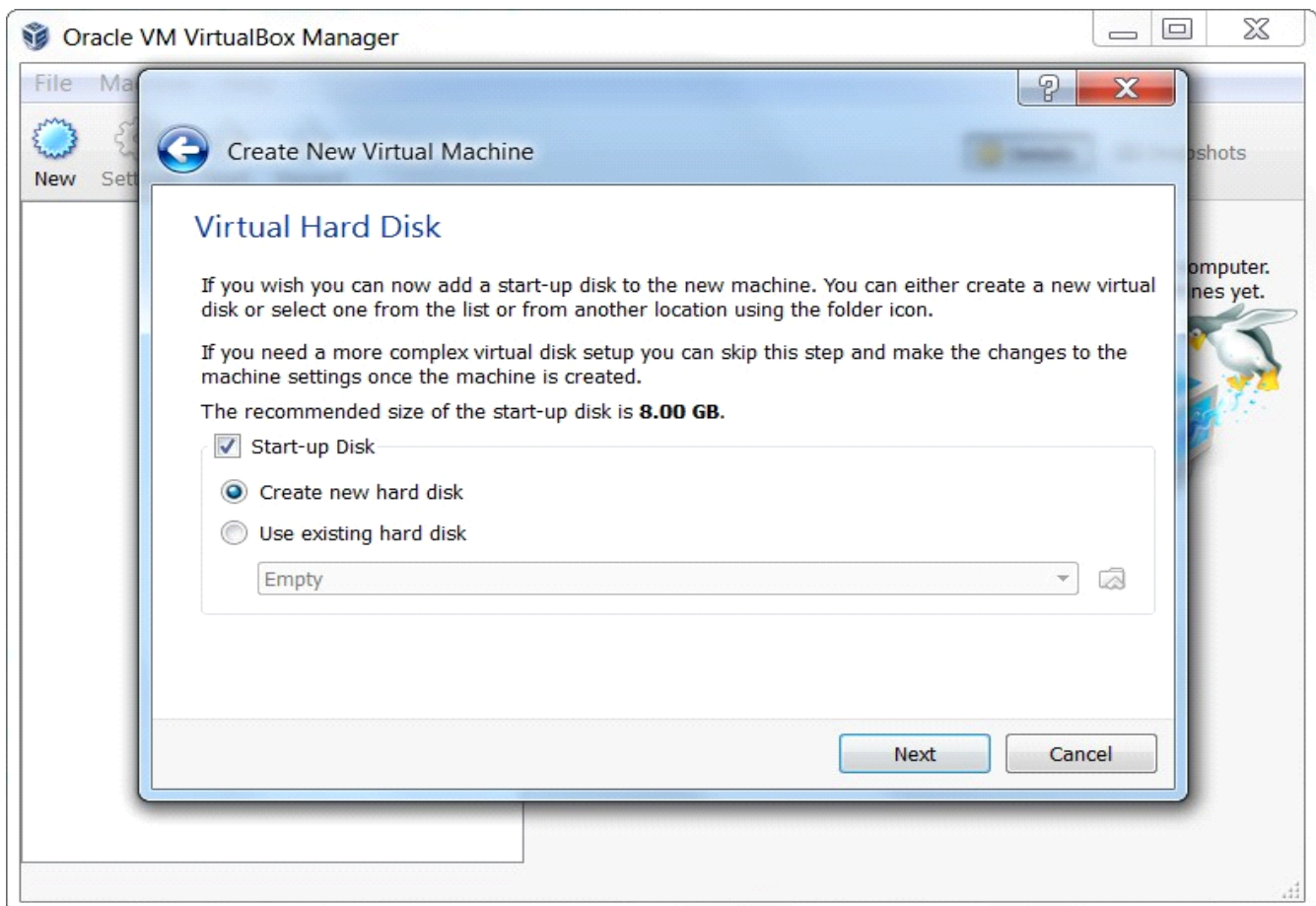
In fereastra «Name and operating system» specifica numele masinii virtuale, selecteaza tipul si versiunea sistemului de instalare care va fi instalat. In cazul acesta – Ubuntu. Apoi apasa «Next».



Specificati cantitatea memoriei RAM alocata masinii virtuale, doar atat cat aveti nevoie, (deoarece memoria va fi “imprumutata” de la sistemul principal, cand masina virtuala va fi lansata). Memoria recomandată este între (minimum) 512MB și 1GB.

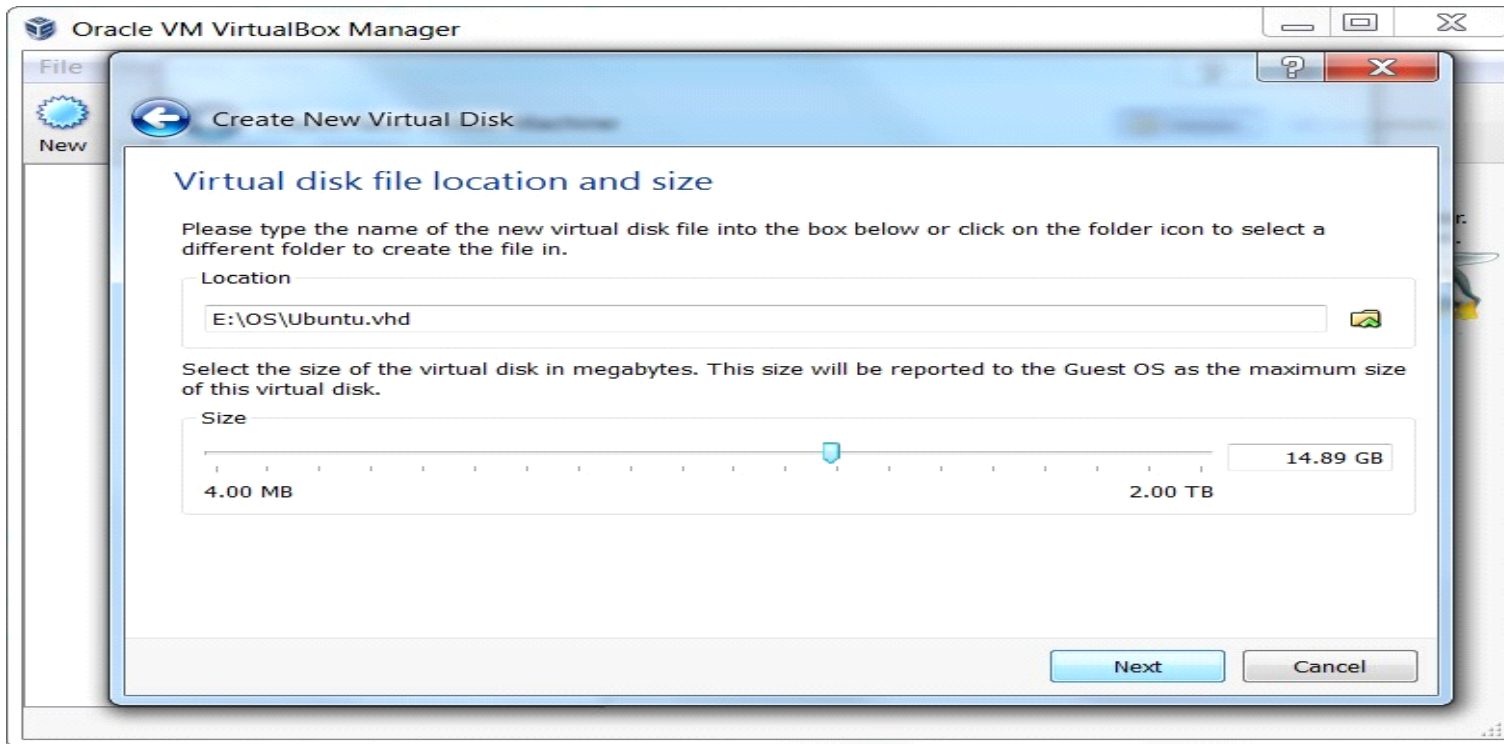


În fereastra următoare selectați «Create a virtual hard disk now».

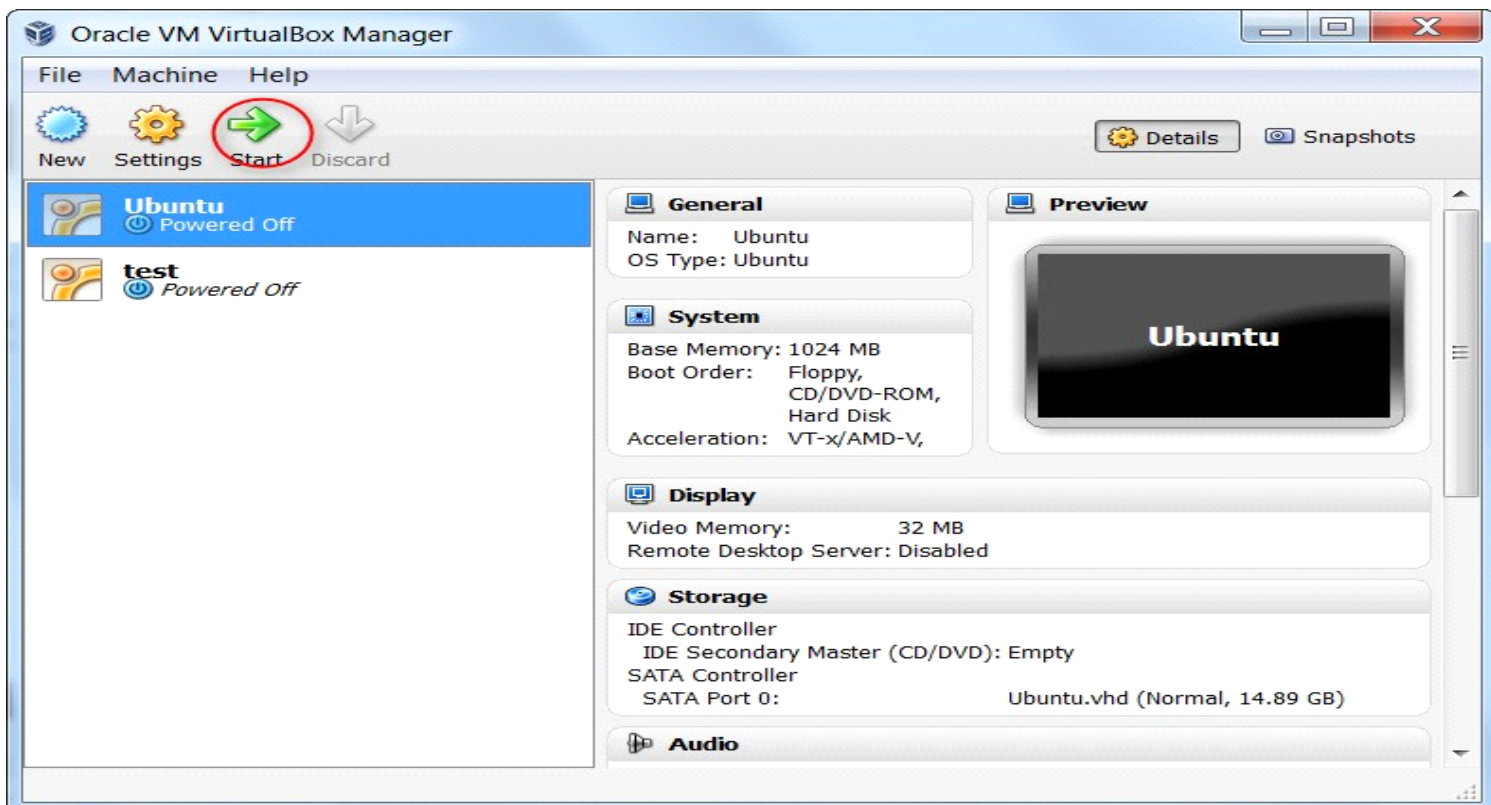




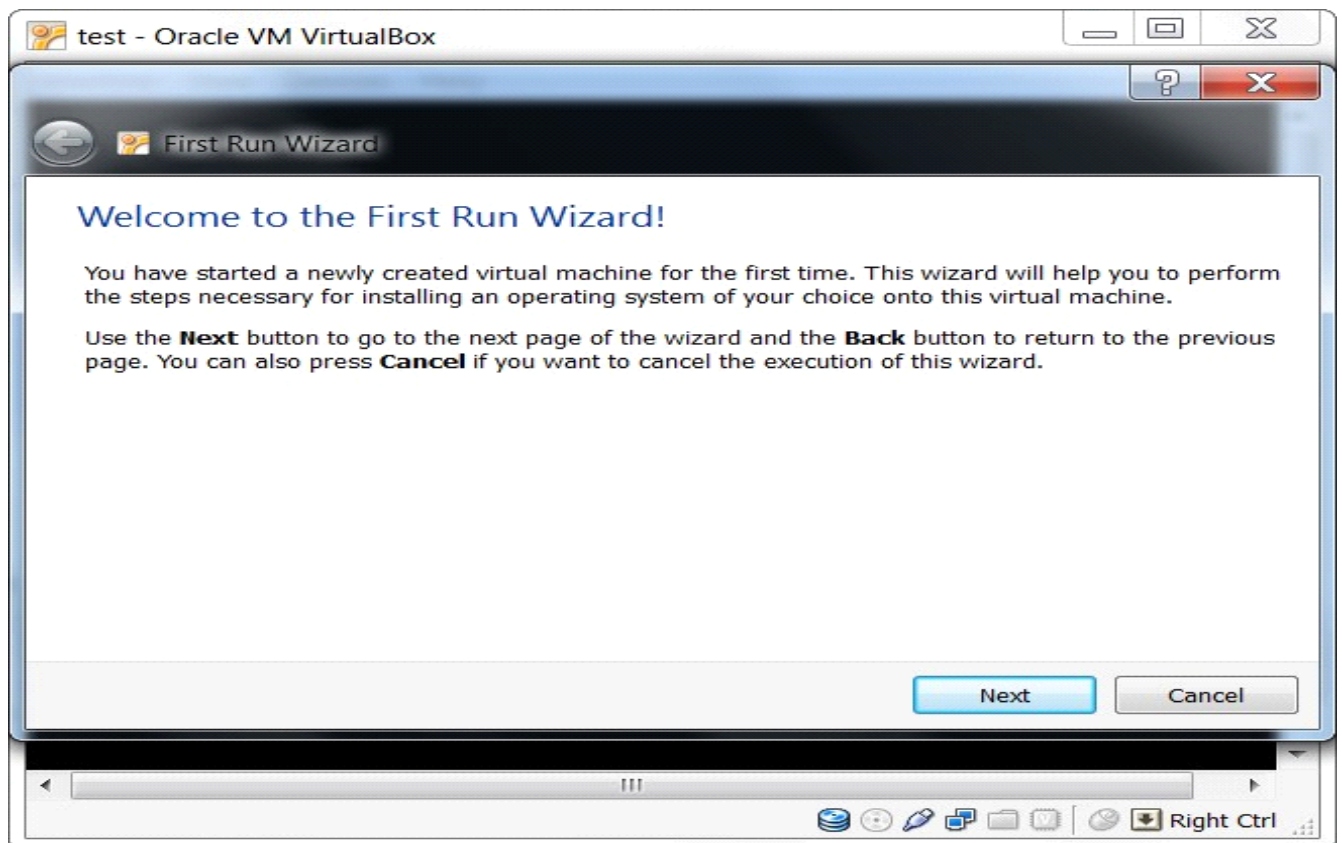
Specifica marimea hard diskului virtual (4,5 GB minimum pentru Ubuntu) si locatia de stocare pe calculator (marimea sa fie suficienta pentru instalarea si rularea sistemului de operare oaspete). Apasa «Create» si asteapta terminarea crearii discului virtual.



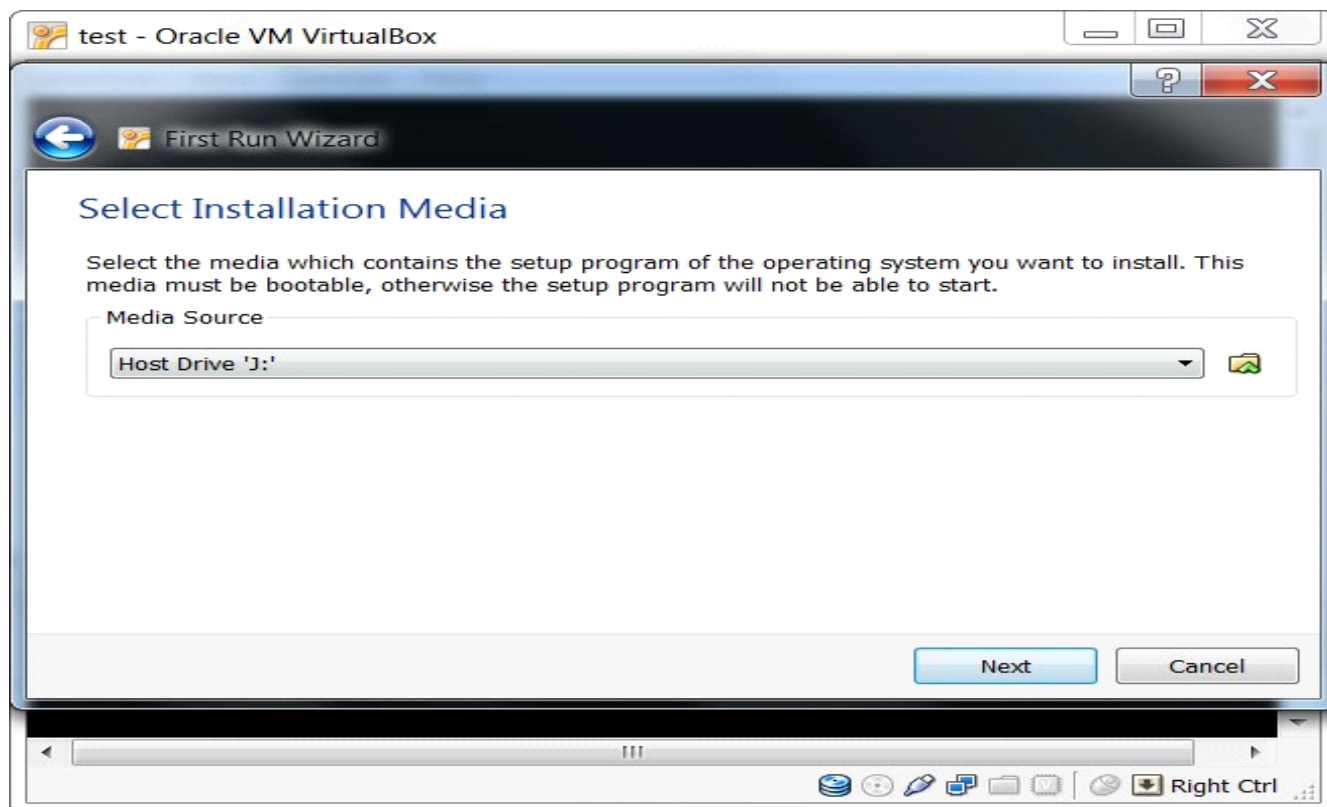
In final porniți mașina virtuală. Pentru a vedea informațiile de configurare, ca si in screenshot, apăsați sageata din dreapta butonului «Machine Tools» si selectați «Details».



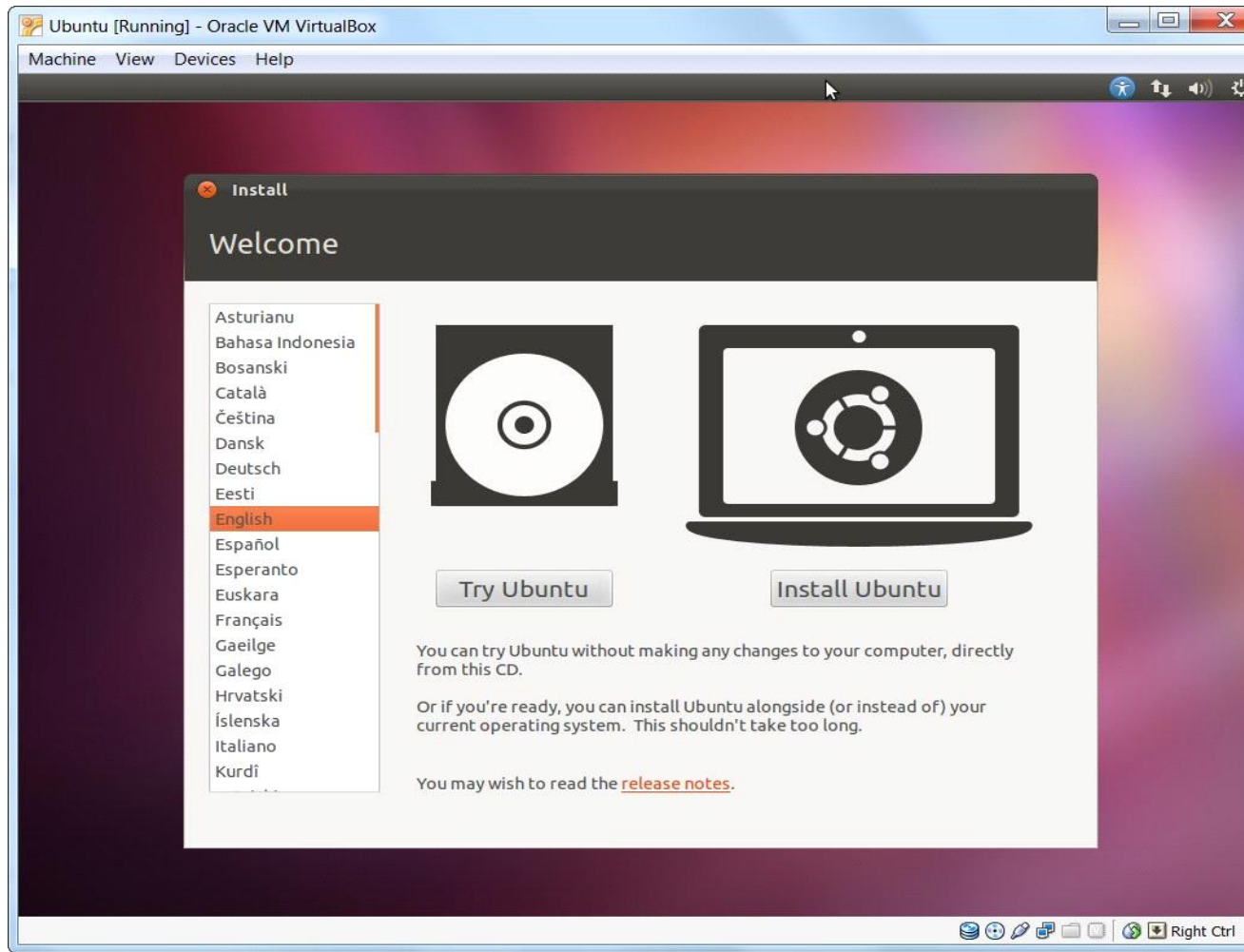
## Instalarea Linux in VirtualBox



- Selectați imaginea Ubuntu pe care ați descărcat-o anterior.



- Așteptați până se încarcă Ubuntu și apoi dați install.



O versiune în limba engleză a modului de instalare a programului VirtualBox (Ubuntu 16.04) poate fi găsită la adresele:

[https://seedsecuritylabs.org/Labs\\_16.04/Documents/SEEDVM\\_VirtualBoxManual.pdf](https://seedsecuritylabs.org/Labs_16.04/Documents/SEEDVM_VirtualBoxManual.pdf)

#### Alte resurse

<http://download.virtualbox.org/virtualbox/UserManual.pdf>

**Tema laborator:** Instalați VirtualBox și creați o mașină virtuală cu o distribuție de Linux (o veți utiliza în laboratoarele următoare). Puteți instala Ubuntu (versiunea 32biți are cerințe mai reduse). Alte opțiuni: Lubuntu, Ubuntu Server, ArchLinux, etc.

Pentru gestiunea mașinilor virtuale dintr-un sistem VirtualBox ne pune la dispoziție suita de comenzi **vboxmanage**. Aceasta poate să:

- listeze toate mașinile virtuale

```
$ vboxmanage list vms
```

- sau doar a celor care sunt pornite în mod curent

```
$ vboxmanage list runningvms
```

- oprirea unei mașini după nume

```
$ vboxmanage controlvm NOME_VM acpipowerbutton
```



- restart-area mașinii

```
$ vboxmanage controlvm NUME_VM reset
```

- pornirea acesteia, folosind numele

```
$ vboxmanage startvm NUME_VM
```

O funcționalitate utilă a unei mașini virtuale este aceea de a-și salva starea curentă, prin trecerea în starea de pauză (pause), în care nu consumă resurse, și revenirea la starea inițială atunci când e nevoie de ea (resume). Comenzile folosite sunt:

- trecerea în starea de pauză

```
$ vboxmanage controlvm NUME_VM pause
```

- revenirea la starea de rulare

```
$ vboxmanage controlvm NUME_VM resume
```

## **biblioteca OpenSSL**

<https://www.openssl.org/>

OpenSSL este o bibliotecă software pentru aplicații care securizează comunicațiile prin rețelele de calculatoare împotriva hackerilor sau pentru situații în care este necesară identificarea părții de la celălalt capăt al terminalului. Este utilizat pe scară largă de serverele de internet, inclusiv de majoritatea site-urilor web HTTPS. (<https://en.wikipedia.org/wiki/OpenSSL>)

<http://gnuwin32.sourceforge.net/packages/openssl.htm>

(openssl for windows)

OpenSSL este o bibliotecă crypto larg folosită care implementează protocoalele SSL and TLS în scopul securizării comunicărilor dintr-o rețea sau între rețele. Biblioteca OpenSSL este folosită de o serie de programe precum Apache Web server, PHP, Postfix etc. OpenSSL furnizează suport pentru o mare varietate de algoritmi criptografici precum ciphers (AES, Blowfish, DES, IDEA etc.), cryptographic hash functions (MD5, MD4, SHA-1, SHA-2 etc.) and public key cryptography (RSA, DSA, Diffie-Hellman key exchange).

[https://www.openssl.org/docs/manmaster/man3/EVP\\_EncryptInit.html](https://www.openssl.org/docs/manmaster/man3/EVP_EncryptInit.html)

pentru utilizarea API OpenSSL EVP este necesară instalarea bibliotecilor openssl:

```
wget https://www.openssl.org/source/openssl-1.0.1q.tar.gz
tar xvf openssl-1.0.1q.tar.gz
cd openssl-1.0.1q
sudo ./config
sudo make
sudo make test
sudo make install
```

How to install OpenSSL, step by step, e.g. from source on Ubuntu 18.04 and CentOS 7.6 servers.

Steps:

- Install Dependencies

- Download OpenSSL Source Code
- Install OpenSSL
  - Compile and Install OpenSSL
  - Configure Link Libraries
  - Configure OpenSSL Binary
- Testing

**Step 1.** Before compiling the OpenSSL library from source, install some package dependencies including the 'build-essential' package on Ubuntu. Update the Ubuntu repository and install package dependencies for software compilation using the apt command below:

```
sudo apt update
sudo apt install build-essential checkinstall zlib1g-dev -y
```

After the installation is complete, go to the next step.

## Step 2 - Download OpenSSL

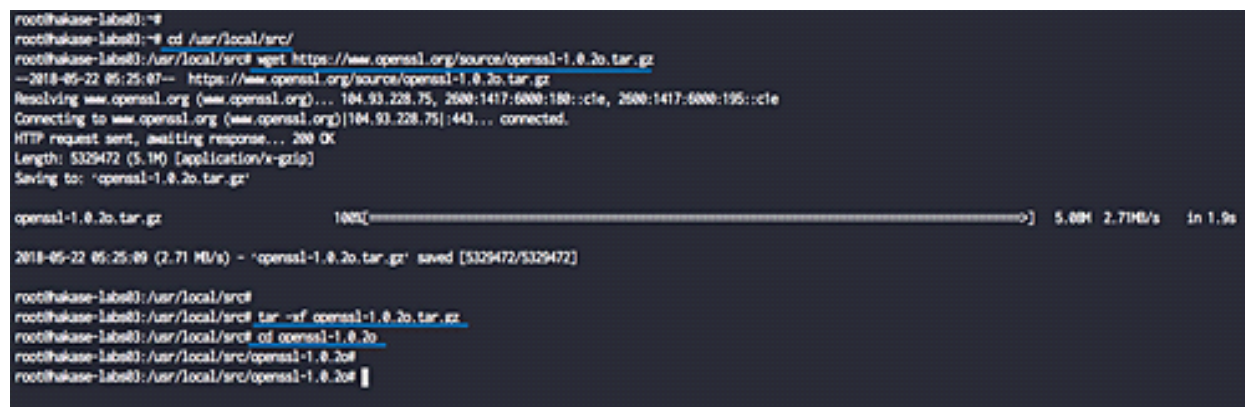
Install the latest stable version of OpenSSL - OpenSSL 1.0.2o. You can download the source code from the [OpenSSL HYPERLINK "https://www.openssl.org/source/"](https://www.openssl.org/source/) site.

Go to the '/usr/local/src' directory and download the OpenSSL source code using wget.

```
cd /usr/local/src/
wget https://www.openssl.org/source/openssl-1.0.2o.tar.gz
```

Next extract the openssl.tar.gz file, and go to the 'openssl' directory.

```
tar -xf openssl-1.0.2o.tar.gz
cd openssl-1.0.2o
```



```
root@hukse-lab08:~#
root@hukse-lab08:~# cd /usr/local/src/
root@hukse-lab08:/usr/local/src# wget https://www.openssl.org/source/openssl-1.0.2o.tar.gz
--2018-05-22 05:25:07-- https://www.openssl.org/source/openssl-1.0.2o.tar.gz
Resolving www.openssl.org (www.openssl.org)... 104.93.228.75, 2000:1417:0000:100::cfe, 2000:1417:0000:195::cfe
Connecting to www.openssl.org (www.openssl.org)|104.93.228.75|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5329472 (5.1M) [application/x-gzip]
Saving to: 'openssl-1.0.2o.tar.gz'

openssl-1.0.2o.tar.gz      100%[=====>] 5.00M  2.71MB/s   in 1.9s

2018-05-22 05:25:09 (2.71 MB/s) - 'openssl-1.0.2o.tar.gz' saved [5329472/5329472]

root@hukse-lab08:/usr/local/src#
root@hukse-lab08:/usr/local/src# tar -xf openssl-1.0.2o.tar.gz
root@hukse-lab08:/usr/local/src# cd openssl-1.0.2o
root@hukse-lab08:/usr/local/src/openssl-1.0.2o#
```

## Step 3 - Install OpenSSL

Before installing the custom OpenSSL version to the system, check the installed version using the command below:

```
openssl version -a
```

The answer on Ubuntu:

[illegible]

Replace the '1.1.0g' version with the latest stable version 1.0.2o.

Install the new OpenSSL version to the specific directory '/usr/local/ssl', and then enable the Link Libraries of OpenSSL, and configure the new binary PATH for OpenSSL.

## Install and Compile OpenSSL

Go to the openssl downloaded directory '/usr/local/src/openssl'

```
cd /usr/local/src/openssl-1.0.2o
```

Configure and compile OpenSSL with commands below:

```
./config --prefix=/usr/local/ssl --openssldir=/usr/local/ssl shared zlib
```

make

```
make test
```

Wait for the OpenSSL compile process.

**Note that**

- `--prefix` and `--openssldir` = Set the output path of the OpenSSL.
- `shared` = force to create a shared library.
- `zlib` = enable the compression using `zlib` library.

When the compile process is complete, install the OpenSSL using the command below:

```
make install
```

See any operating system documentation and manpages about shared libraries for your version of UNIX. The following manpages may be helpful: ld(1), ld.so(1), ld.so.1(1) [Solaris], dld.sl(1) [HP], ldd(1), crle(1) [Solaris], pldd(1) [Solaris], ldconfig(8) [Linux], chatr(1) [HP].

```
cp libcrypto.pc /usr/local/ssl/lib/pkgconfig
chmod 644 /usr/local/ssl/lib/pkgconfig/libcrypto.pc
cp libssl.pc /usr/local/ssl/lib/pkgconfig
chmod 644 /usr/local/ssl/lib/pkgconfig/libssl.pc
cp openssl.pc /usr/local/ssl/lib/pkgconfig
chmod 644 /usr/local/ssl/lib/pkgconfig/openssl.pc
root@hakase-ubuntu: /usr/local/src/openssl-1.0.2o#
root@hakase-ubuntu: /usr/local/src/openssl-1.0.2o#
```

OpenSSL is installed in the '/usr/local/ssl' directory.

```
[root@hakase-centos ~]#  
[root@hakase-centos ~]# cd /usr/local/ssl/  
[root@hakase-centos ssl]# ll  
total 12  
drwxr-xr-x. 2 root root   37 May 24 16:08 bin  
drwxr-xr-x. 2 root root    6 May 24 16:08 certs  
drwxr-xr-x. 3 root root   21 May 24 16:08 include  
drwxr-xr-x. 4 root root  159 May 24 16:08 lib  
drwxr-xr-x. 6 root root   54 May 24 16:06 man  
drwxr-xr-x. 2 root root  103 May 24 16:08 misc  
-rw-r--r--. 1 root root 10835 May 24 16:08 openssl.cnf  
drwxr-xr-x. 2 root root    6 May 24 16:08 private  
[root@hakase-centos ssl]#  
[root@hakase-centos ssl]#
```

### Configure Link Libraries

Next, we will configure the shared libraries for OpenSSL. The new OpenSSL binary will load library files from the '/usr/local/ssl/lib' directory.

Go to the '/etc/ld.so.conf.d' directory and create new configuration file 'openssl-1.0.2o.conf'

```
cd /etc/ld.so.conf.d/  
vim openssl-1.0.2o.conf
```

Paste the openssl library path directory.

```
/usr/local/ssl/lib
```

Save and exit.

Now reload the dynamic link using the command below.

```
sudo ldconfig -v
```

And you will see the OpenSSL libraries on the '/usr/local/ssl/lib' directory has been loaded.

```
root@hakase-ubuntu:~#  
root@hakase-ubuntu:~# cd /etc/ld.so.conf.d/  
root@hakase-ubuntu:/etc/ld.so.conf.d# vim openssl-1.0.2o.conf  
root@hakase-ubuntu:/etc/ld.so.conf.d#  
root@hakase-ubuntu:/etc/ld.so.conf.d# sudo ldconfig -v  
/sbin/ldconfig.real: Can't stat /usr/local/lib/x86_64-linux-gnu: No such file or directory  
/sbin/ldconfig.real: Path '/lib/x86_64-linux-gnu' given more than once  
/sbin/ldconfig.real: Path '/usr/lib/x86_64-linux-gnu' given more than once  
/usr/lib/x86_64-linux-gnu/libfakeroot:  
    libfakeroot-0.so -> libfakeroot-tcp.so  
/usr/local/lib:  
/usr/local/ssl/lib:  
    libcrypto.so.1.0.0 -> libcrypto.so.1.0.0  
    libssl.so.1.0.0 -> libssl.so.1.0.0  
/lib/x86_64-linux-gnu:
```



## Configure OpenSSL Binary

Replace the default openssl binary '/usr/bin/openssl or /bin/openssl' with the new version '/usr/local/ssl/bin/openssl'.

### On Ubuntu 18.04 LTS

Backup the binary files.

```
mv /usr/bin/c_rehash /usr/bin/c_rehash.BEKUP
mv /usr/bin/openssl /usr/bin/openssl.BEKUP
```

Edit the '/etc/environment' file using [vim](#).

```
vim /etc/environment
```

Now add the new OpenSSL binary directory as below

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/local/ssl/bin"
```

Save and exit.

Reload the environment file and test the new updated binary PATH.

```
source /etc/environment
echo $PATH
```

Now check again the OpenSSL binary file.

```
which openssl
```

You will get the result as below.



```
root@hakase-ubuntu:~#
root@hakase-ubuntu:~# which openssl
/usr/bin/openssl
root@hakase-ubuntu:~#
root@hakase-ubuntu:~# mv /usr/bin/c_rehash /usr/bin/c_rehash.BEKUP
root@hakase-ubuntu:~# mv /usr/bin/openssl /usr/bin/openssl.BEKUP
root@hakase-ubuntu:~#
root@hakase-ubuntu:~# vim /etc/environment
root@hakase-ubuntu:~#
root@hakase-ubuntu:~# source /etc/environment
root@hakase-ubuntu:~#
root@hakase-ubuntu:~# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/local/ssl/bin
root@hakase-ubuntu:~#
root@hakase-ubuntu:~# which openssl
/usr/local/ssl/bin/openssl
root@hakase-ubuntu:~#
root@hakase-ubuntu:~#
```

The binary path of OpenSSL for Ubuntu has been updated.

Make the openssl.sh file executable.

```
chmod +x /etc/profile.d/openssl.sh
```

Load the OpenSSL environment and check the PATH bin directory using commands below.

```
source /etc/profile.d/openssl.sh
echo $PATH
```

Now check the OpenSSL file.

```
which openssl
```

You will get the result as below.

```
[root@hakase-centos ~]#  
[root@hakase-centos ~]# which openssl  
/bin/openssl  
[root@hakase-centos ~]#  
[root@hakase-centos ~]# mv /bin/openssl /bin/openssl.BEKUP  
[root@hakase-centos ~]#  
[root@hakase-centos ~]# vim /etc/profile.d/openssl.sh  
[root@hakase-centos ~]# chmod +x /etc/profile.d/openssl.sh  
[root@hakase-centos ~]#  
[root@hakase-centos ~]# source /etc/profile.d/openssl.sh  
[root@hakase-centos ~]# echo $PATH  
/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/ssl/bin  
[root@hakase-centos ~]#  
[root@hakase-centos ~]# which openssl  
/usr/local/ssl/bin/openssl  
[root@hakase-centos ~]#  
[root@hakase-centos ~]# █
```

The binary path for OpenSSL on CentOS has been updated.

## Step 4 - Testing

Test the OpenSSL new version using the following command.

```
openssl version -a
```

### The result

[illegible]

The new latest stable version of OpenSSL has been installed from source on Linux Ubuntu 18.04.

## Reference

<https://wiki.openssl.org/>

<https://www.howtoforge.com/tutorial/how-to-install-openssl-from-source-on-linux/>