VirtualBox, Vagrant

Lecture 2.2

Module 2. Virtualization and Cloud Basic

Andrii Kostromytskyi

Agenda

- VirtualBox
- Vagrant
- Q&A

VIRTUALBOX

VirtualBox

The techniques and capabilities that VirtualBox provides can be used for the following scenarios:

OS support. In VirtualBox, it is possible to run programs written for another operating system (for example, programs for Windows on Linux systems) without the need to load this OS. You can also install "old" OSs, such as DOS or OS / 2, which cannot work on your hardware due to its "advanced" nature.

Infrastructure consolidation Virtualization can significantly reduce hardware and electricity costs. The power of systems provided by modern hardware is rarely fully utilized; a typical server usually uses half of its theoretical power. So, instead of using several physical computers that are only partially loaded, you can run several virtual machines on powerful host computers and distribute the load between them. VirtualBox can work as a simple VirtualBox Remote Desktop Protocol (VRDP - Remote Desktop Protocol) server with USB support. This allows you to accompany the entire software infrastructure of the enterprise only on a few RDP servers (terminal server), and in fact, client systems only need to be a VRDP client (thin client).

Testing and recovery in emergency situations. After installation and configuration, the virtual machine and its virtual hard disk can be considered a "container", which can be "frozen", "woken up", copied and transferred to other computers. In addition to this, using the VirtualBox mechanism called “snapshots”, you can save the state of the virtual machine and roll back to this state, if necessary. You can freely experiment with the computing environment. If something goes wrong (for example, after improper software installation or infection of the guest OS with a virus), you can easily switch back to the previous snapshot of the system without performing frequent backups and restoring them.

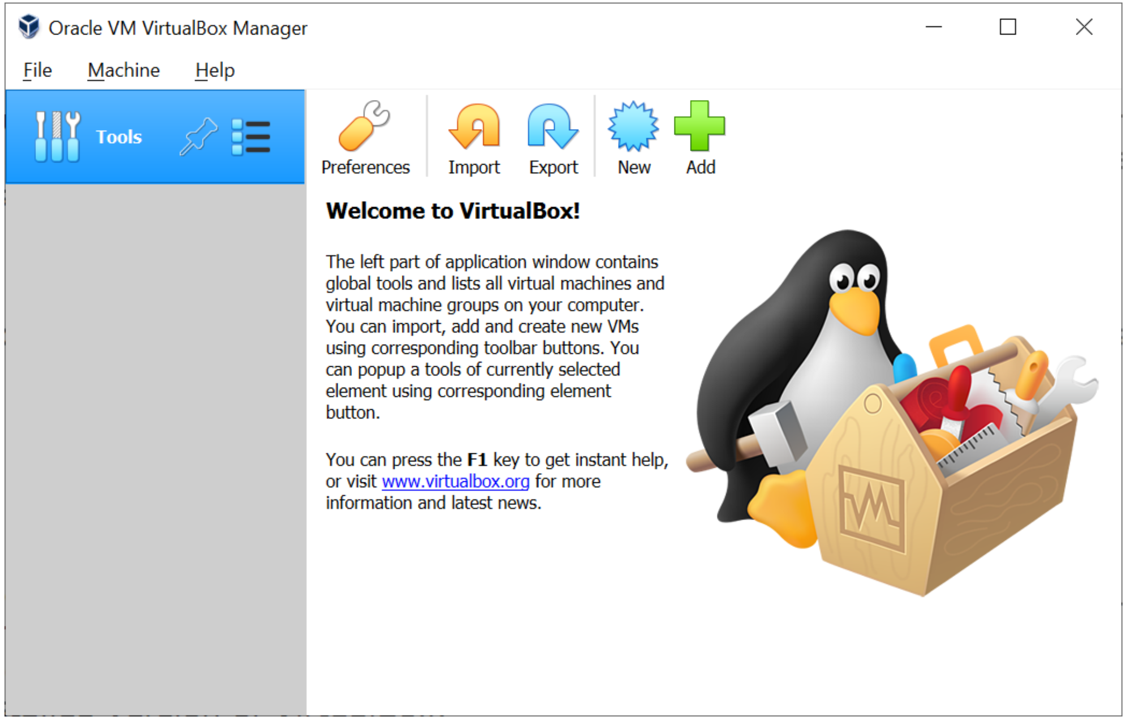
Key Terms, GUI

- Host operating system (host OS)
- Guest operating system (guest OS)
- Virtual machine (VM)
- Guest Additions.

VirtualBox 6.1.10

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

https://www.virtualbox.org/manual/UserManual.html



Welcome to VirtualBox!

The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can import, add and create new VMs using corresponding toolbar buttons. You can popup a tools of currently selected element using corresponding element button.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for more information and latest news.

Pros and cons

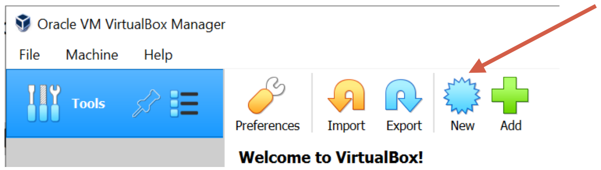
What good

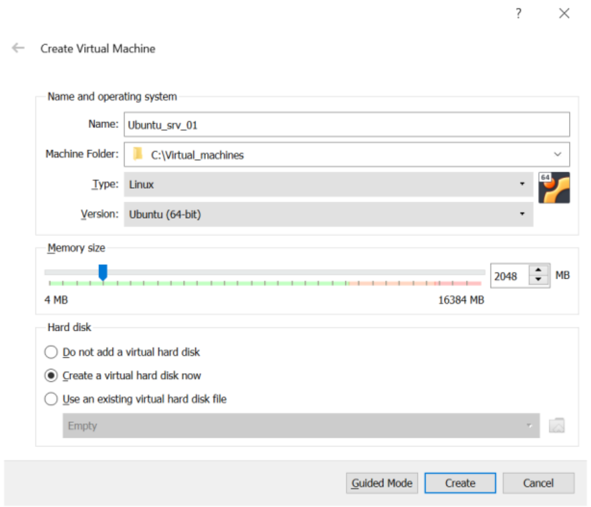
- Ease of use
- An excellent guide detailing all aspects of VirtualBox.
- The presence of excellent graphical, console and web-based interface
- The ability to provide access to the guest OS console about the RDP protocol
- The convenience of use
- The complete VirtualBox user guide is available on the manufacturer’s website.

What is bad

- Lack of good web muzzles. The existing web-interface, despite the fact that it allows you to perform most of the required actions with virtual machines (create, take pictures, delete, create virtual networks), and is implemented very efficiently, has the following limitations:
    - It does not allow assigning rights to virtual machines (granting developers access to the list only to their machines)
    - Cannot manage multiple physical servers from one control panel
    - Does not display server load statistics
    - It is written by third-party developers who are not related to VirtualBox in their free time, which causes concern about the possibility of stopping its development
- Slower operation with a large number of running virtual machines compared to KVM.
- License clause

Create VM in VirtualBox





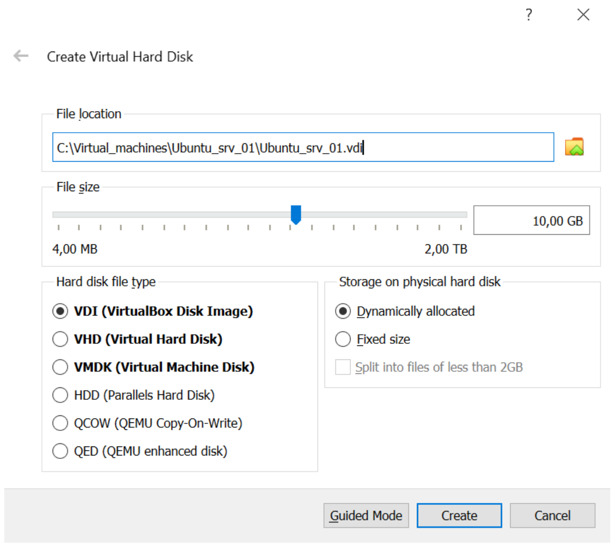
Name and operating system

Name:	Ubuntu_srv_01
Machine Folder:	C:\Virtual_machines
Type:	Linux
Version:	Ubuntu (64-bit)

Memory size: 2048 MB (4 MB – 16384 MB)

Hard disk

- Do not add a virtual hard disk
- [✓ Create a virtual hard disk now]
- Use an existing virtual hard disk file: Empty



File location: C:\Virtual_machines\Ubuntu_srv_01\Ubuntu_srv_01.vdi

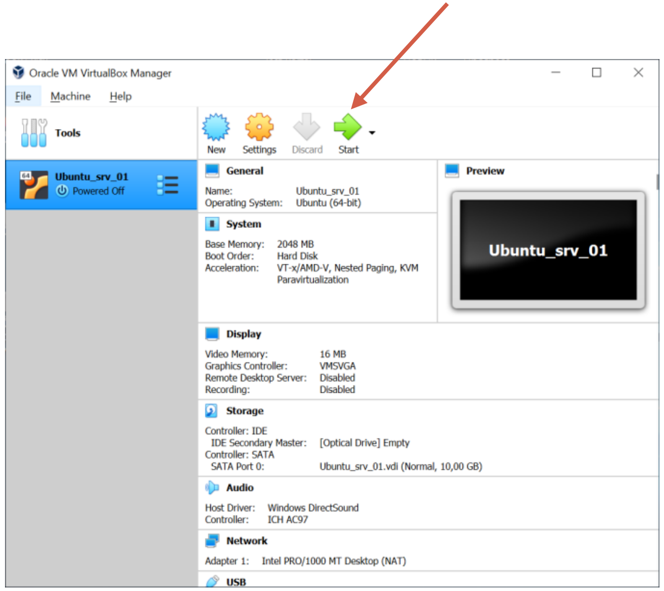
File size: 10,00 GB (4,00 MB – 2,00 TB)

Hard disk file type

- [✓ VDI (VirtualBox Disk Image)]
- VHD (Virtual Hard Disk)
- VMDK (Virtual Machine Disk)
- HDD (Parallels Hard Disk)
- QCOW (QEMU Copy-On-Write)
- QED (QEMU enhanced disk)

Storage on physical hard disk

- [✓ Dynamically allocated]
- Fixed size
- Split into files of less than 2GB



General

Name:	Ubuntu_srv_01
Operating System:	Ubuntu (64-bit)

System

Base Memory:	2048 MB
Boot Order:	Hard Disk
Acceleration:	VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Display

Video Memory:	16 MB
Graphics Controller:	VMSVGA
Remote Desktop Server:	Disabled
Recording:	Disabled

Storage

Controller: IDE	
IDE Secondary Master:	[Optical Drive] Empty
Controller: SATA	
SATA Port 0:	Ubuntu_srv_01.vdi (Normal, 10,00 GB)

Audio

Host Driver:	Windows DirectSound
Controller:	ICH AC97

Network

Adapter 1:	Intel PRO/1000 MT Desktop (NAT)

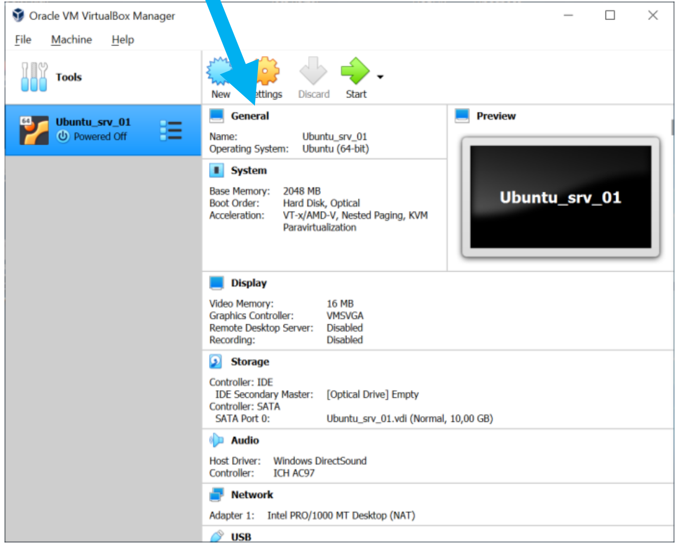
USB

Configuring Public Folders in VirtualBox

1 Setting

2 Shared Folders

3 Adds new Shared Folders



General
Name: Ubuntu_srv_01
Operating System: Ubuntu (64-bit)

System
Base Memory: 2048 MB
Boot Order: Hard Disk, Optical
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

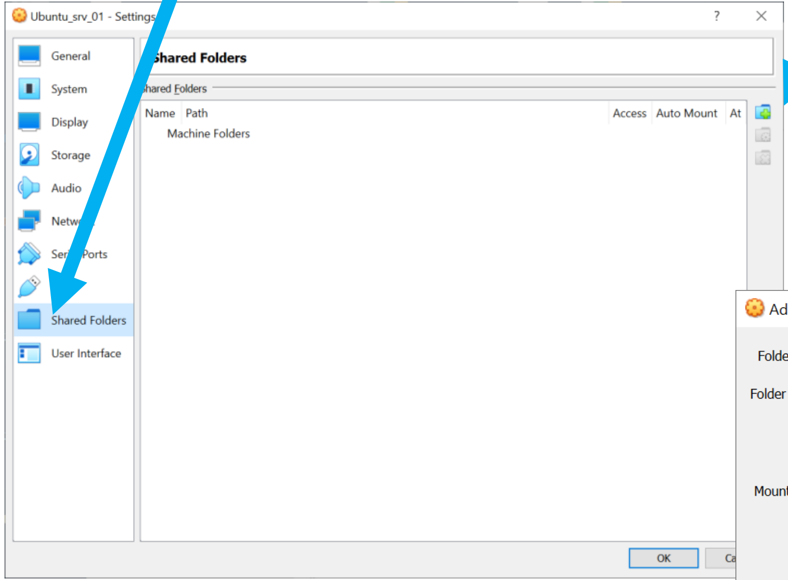
Display
Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled

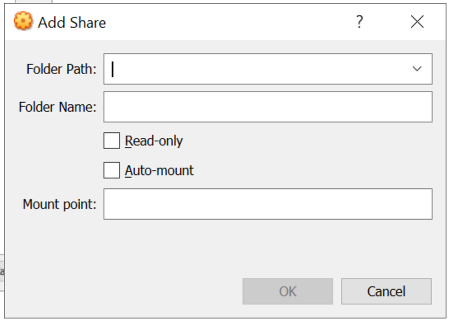
Storage
Controller: IDE
IDE Secondary Master: [Optical Drive] Empty
Controller: SATA
SATA Port 0: Ubuntu_srv_01.vdi (Normal, 10,00 GB)

Audio
Host Driver: Windows DirectSound
Controller: ICH AC97

Network
Adapter 1: Intel PRO/1000 MT Desktop (NAT)

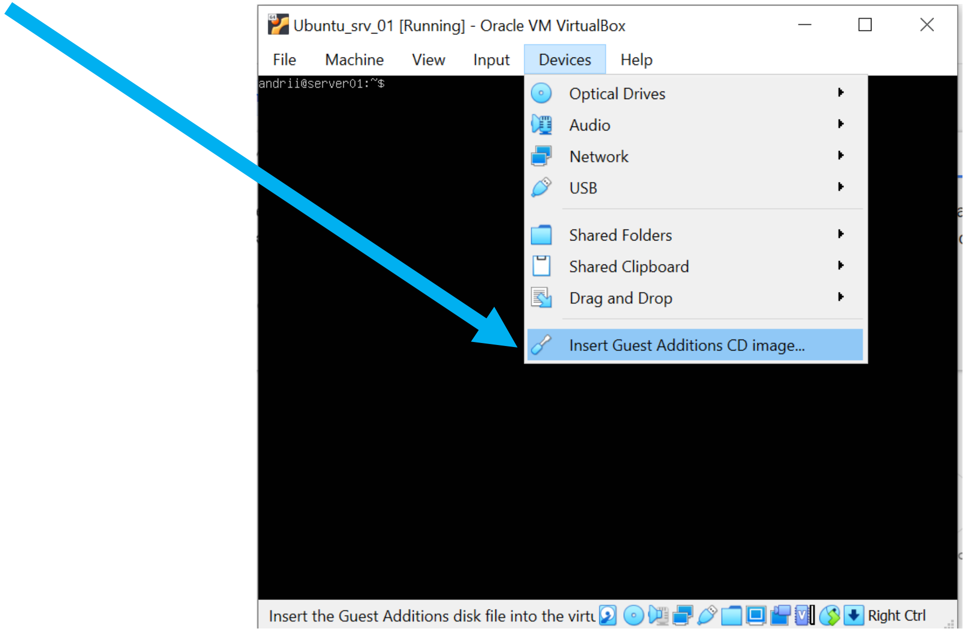
USB





Setting up shared folders in VirtualBox on a guest OS

1 In the virtual machine window, select “Devices” and click on “Insert Guest CD Additions CD image.”



```
andrii@server01:~$
```

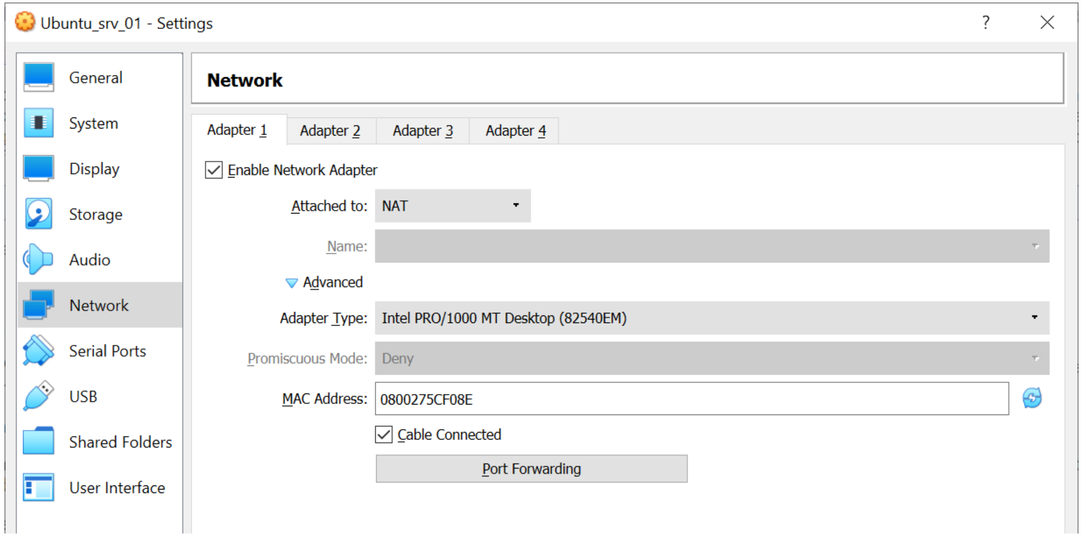
Insert the Guest Additions disk file into the virtu

2 add the user to the vboxsf group in the terminal and from root execute the command

```
sudo adduser myuser vboxsf
```

Introduction to networking modes

- Not attached.
- Network Address Translation (NAT)
- NAT Network.
- Bridged networking.
- Internal networking
- Host-only networking
- Generic networking


Ubuntu_srv_01 - Settings

Network

☑ Enable Network Adapter

Attached to:	NAT
Name:	
Adapter Type:	Intel PRO/1000 MT Desktop (82540EM)
Promiscuous Mode:	Deny
MAC Address:	0800275CF08E

☑ Cable Connected

Mode	VM→Host	VM←Host	VM1↔VM2	VM→Net/LAN	VM←Net/LAN
Host-only	+	+	+	–	–
Internal	–	–	+	–	–
Bridged	+	+	+	+	+
NAT	+	Port forward	–	+	Port forward
NATservice	+	Port forward	+	+	Port forward

https://www.virtualbox.org/manual/UserManual.html#networkingdetails chapter 6

CLI VBoxManage

Using the CLI and VBoxManage, you can implement all the functions available in the GUI and some not available in the GUI

1. VBoxManage always used with subcommands for example “list”, “createvm”, “startvm”
2. Most subcommands require a specific VM after the subcommand..

```
VBoxManage startvm ‘Windows XP’
VBoxManage startvm 670e746d-abea-4ba6-ad02-2a3b043810a5
```

list	import	storagectl
showvminfo	export	createmedium
createvm	startvm	guestcontrol
modifyvm	controlvm	metrics
clonevm	snapshot	natnetwork

https://www.virtualbox.org/manual/ch08.html

VAGRANT

Vagrant

Vagrant is a tool that allows you to timely create a development or testing environment that is isolated on a virtual machine. For this, different providers are used: VirtualBox, VMWare, etc.

Basically Vagrant does the following:
- Creates a virtual machine based on predefined settings (boxes)
- allows you to change the properties of a virtual machine;
- Configures network interfaces
- Shares folders between the host and the virtual machine
- indicates the host name of the guest machine;
- security software on the machine using a set of assistants: Puppet, Chef, Shell, others;

After the machine is up and running, you can:
- login via ssh;
- stop or pause;
- reload.

Vagrant first step

1. Installation Vagrant https://www.vagrantup.com/docs/installation/index.html
2. You can download Boxes from
   http://www.vagrantbox.es/ https://app.terraform.io/boxes/search
3. It is very important to use Vagrant boxes that matches the version of your virtualization software, otherwise some of the configuration settings and execution of Vagrant will fail.
4. Install the box after downloading locally, you can:

vagrant box add {title} {url}

```
vagrant box add centos64_puppet_32bit http://developer.nrel.gov/downloads/vagrant-
boxes/CentOS-6.4-i386-v20131103.box
```

```
mkdir C:\Temp\Vagrant
vagrant init centos64_puppet_32bit
```

This command above will create a Vagrantfile that is based on Ruby and describes the necessary configuration.

Vagrantfile

After configuration, you can run the command

```
vagrant up
```

```
# Vagrantfile

# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure(2) do |config|
  config.vm.box = "centos64_puppet_32bit"

  config.ssh.username = "vagrant"
  config.ssh.password = "vagrant"

  # Disable automatic box update checking. If you disable this, then
  # boxes will only be checked for updates when the user runs
  # `vagrant box outdated`. This is not recommended.
  # config.vm.box_check_update = false

  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine. In the example below,
  # accessing "localhost:8080" will access port 80 on the guest machine.
  config.vm.network "forwarded_port", guest: 80, host: 8080

  # Create a private network, which allows host-only access to the machine
  # using a specific IP.
  config.vm.network "private_network", ip: "192.168.34.16"
```

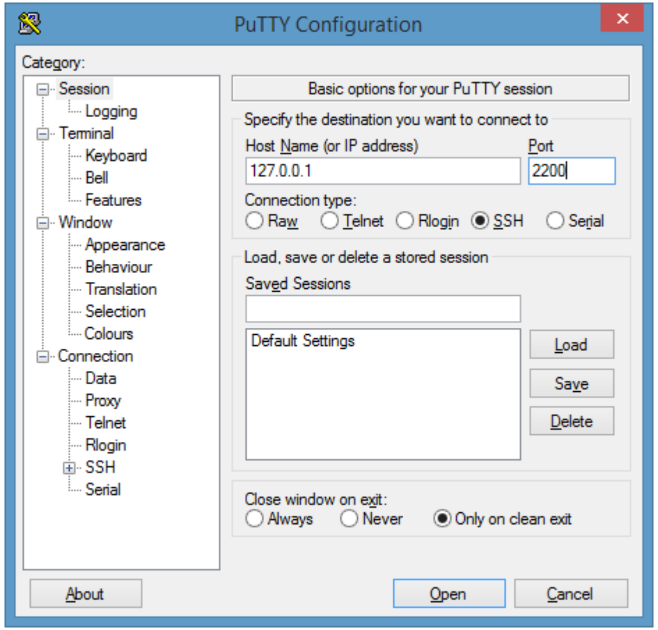
Vagrantfile

```
  # Create a public network, which generally matched to bridged network.
  # Bridged networks make the machine appear as another physical device on
  # your network.
  # config.vm.network "public_network"

  # Share an additional folder to the guest VM. The first argument is
  # the path on the host to the actual folder. The second argument is
  # the path on the guest to mount the folder. And the optional third
  # argument is a set of non-required options.
  config.vm.synced_folder "./data", "/vagrant_data"

  # Provider-specific configuration so you can fine-tune various
  # backing providers for Vagrant.
  # config.vm.provider "virtualbox" do |vb|
  #   # Display the VirtualBox GUI when booting the machine
  #   vb.gui = true
  #
  #   # Customize the amount of memory on the VM:
  #   vb.memory = "1024"
  # end

  # Enable provisioning with a shell script.
  config.vm.provision "shell", inline: <<-SHELL
      sudo yum install mysql-server -y
      sudo yum install httpd mod_ssl -y
      sudo /usr/sbin/apachectl start
      sudo iptables -I INPUT -p tcp --dport 80 -j ACCEPT
  SHELL
end
```

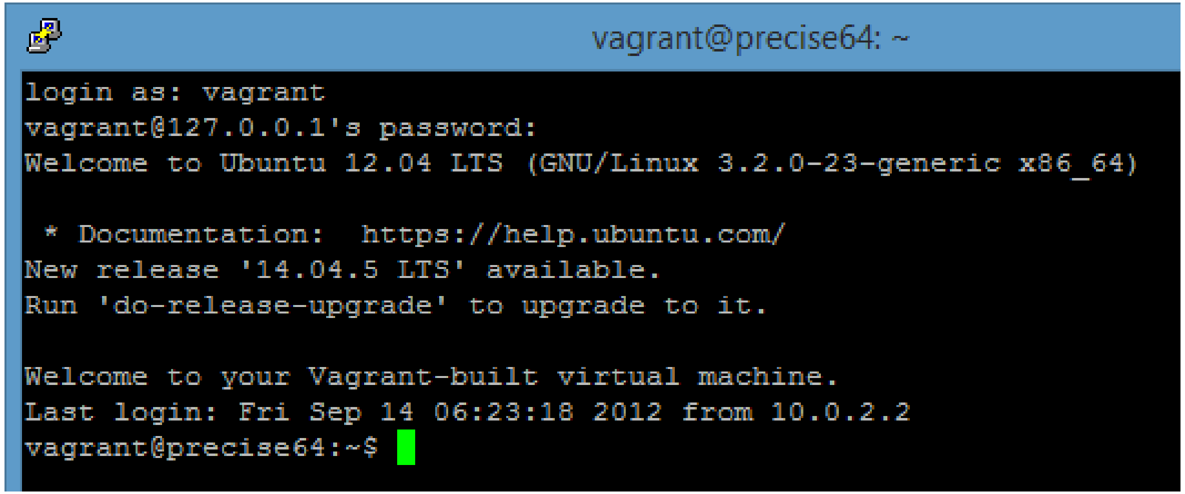
vagrant up

```
PS E:\vagrant_test2> vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'hashicorp/precise64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'hashicorp/precise64' is up to date...
==> default: Setting the name of the VM: vagrant_test2_default_1537902296537_28828
==> default: Fixed port collision for 22 => 2222. Now on port 2200.
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2200 (host) (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2200
    default: SSH username: vagrant
    default: SSH auth method: private key
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
    default: The guest additions on this VM do not match the installed version of
    default: VirtualBox! In most cases this is fine, but in rare cases it can
    default: prevent things such as shared folders from working properly. If you see
    default: shared folder errors, please make sure the guest additions within the
    default: virtual machine match the version of VirtualBox you have installed on
    default: your host and reload your VM.
    default:
    default: Guest Additions Version: 4.2.0
    default: VirtualBox Version: 5.2
==> default: Mounting shared folders...
    default: /vagrant => E:/vagrant_test2
```

Connection to VM. Off



Specify the destination you want to connect to
Host Name (or IP address): 127.0.0.1
Port: 2200
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Close window on exit: Always Never [✓ Only on clean exit]



```
vagrant@precise64: ~
login as: vagrant
vagrant@127.0.0.1's password:
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '14.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Welcome to your Vagrant-built virtual machine.
Last login: Fri Sep 14 06:23:18 2012 from 10.0.2.2
vagrant@precise64:~$
```

```
PS E:\vagrant_test2> vagrant halt
==> default: Attempting graceful shutdown of VM...
PS E:\vagrant_test2> vagrant destroy
    default: Are you sure you want to destroy the 'default' VM? [y/N] y
==> default: Destroying VM and associated drives...
PS E:\vagrant_test2>
```

Q&A

Thank you!