(CSC-05) Lab 2 - Parte A: Virtualbox & Vagrant

Neste relatório serão descritos os passos realizados ao seguir o roteiro disponível para o *Laboratório 2 - Parte A: Virtualbox e Vagrant*.

A execução do laboratório foi gravada no formato *asciinema* (gravação do terminal, sem intervalo entre comandos). Para cada subseção à seguir, serão disponibilizados os links referentes às gravações.

Parte I: Introdução

Gravação:

- Asciinema: https://asciinema.org/a/359506

Ambiente

O Laboratório foi executado no seguinte ambiente:

- a
- b
- c

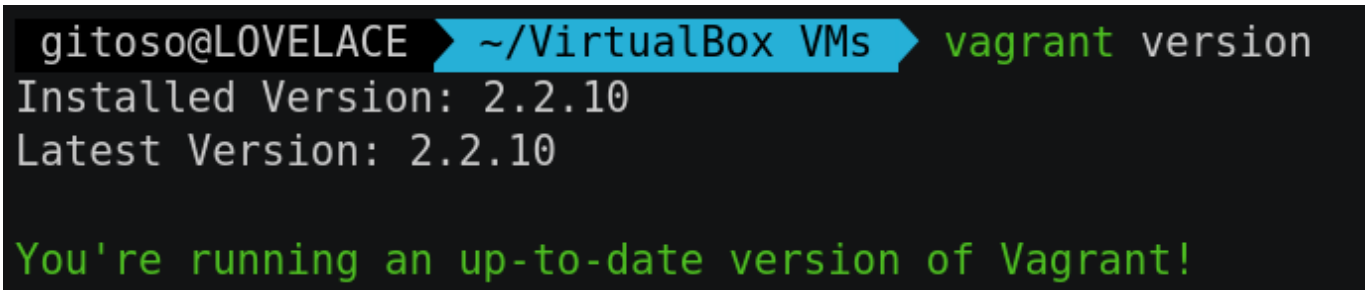
Execução passo-a-passo:

A seguir, a execução dos passos do laboratório será seguida por *screenshots* mostrando o resultado obtido para cada passo.

3) Verificar qual a versão do vagrant está instalada (no powershell ou cmder)

```
# vagrant version
```

No caso executei o Laboratório em um ambiente Linux (Arch Linux x64) e portanto utilizei uma linha de comando para o ambiente Linux.



4) Verificar as “boxes” instaladas

```
# vagrant box list
```

```
gitoso@LOVELACE ~/VirtualBox VMs vagrant box list
There are no installed boxes! Use `vagrant box add` to add some.
```

5) Criar um diretório com o nome “ubuntu18” e entrar nesse diretório

```
# cd Documentos
# mkdir ubuntu18
# cd ubuntu18
```

```
gitoso@LOVELACE ~/VirtualBox VMs mkdir ubuntu18
gitoso@LOVELACE ~/VirtualBox VMs cd ubuntu18
```

6) Criar um arquivo de inicialização do Vagrantfile

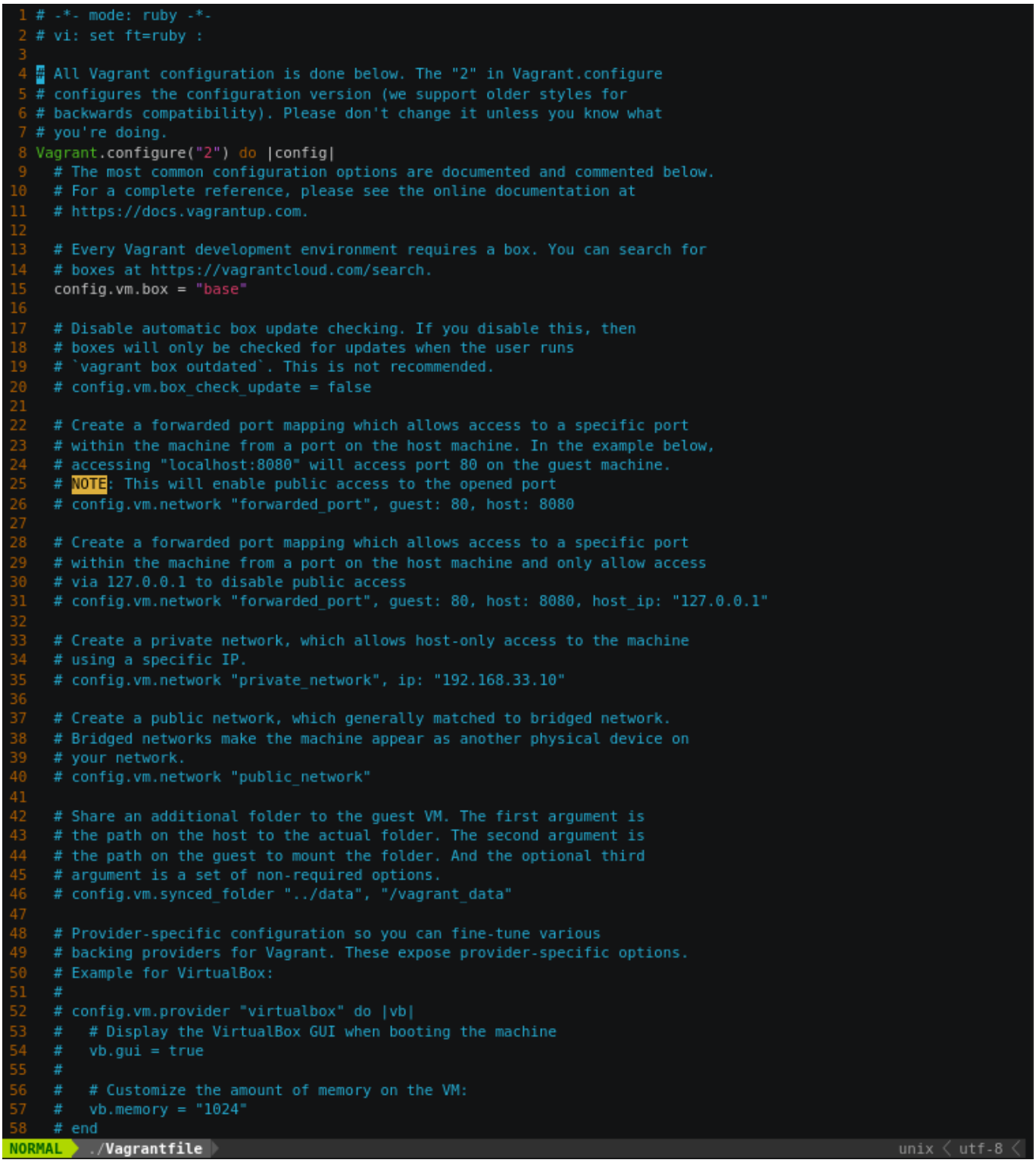
```
# vagrant init
```

```
gitoso@LOVELACE ~/VirtualBox VMs/ubuntu18 vagrant init
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
```

7) Abrir em um editor de texto o arquivo Vagrantfile criado, entender

---

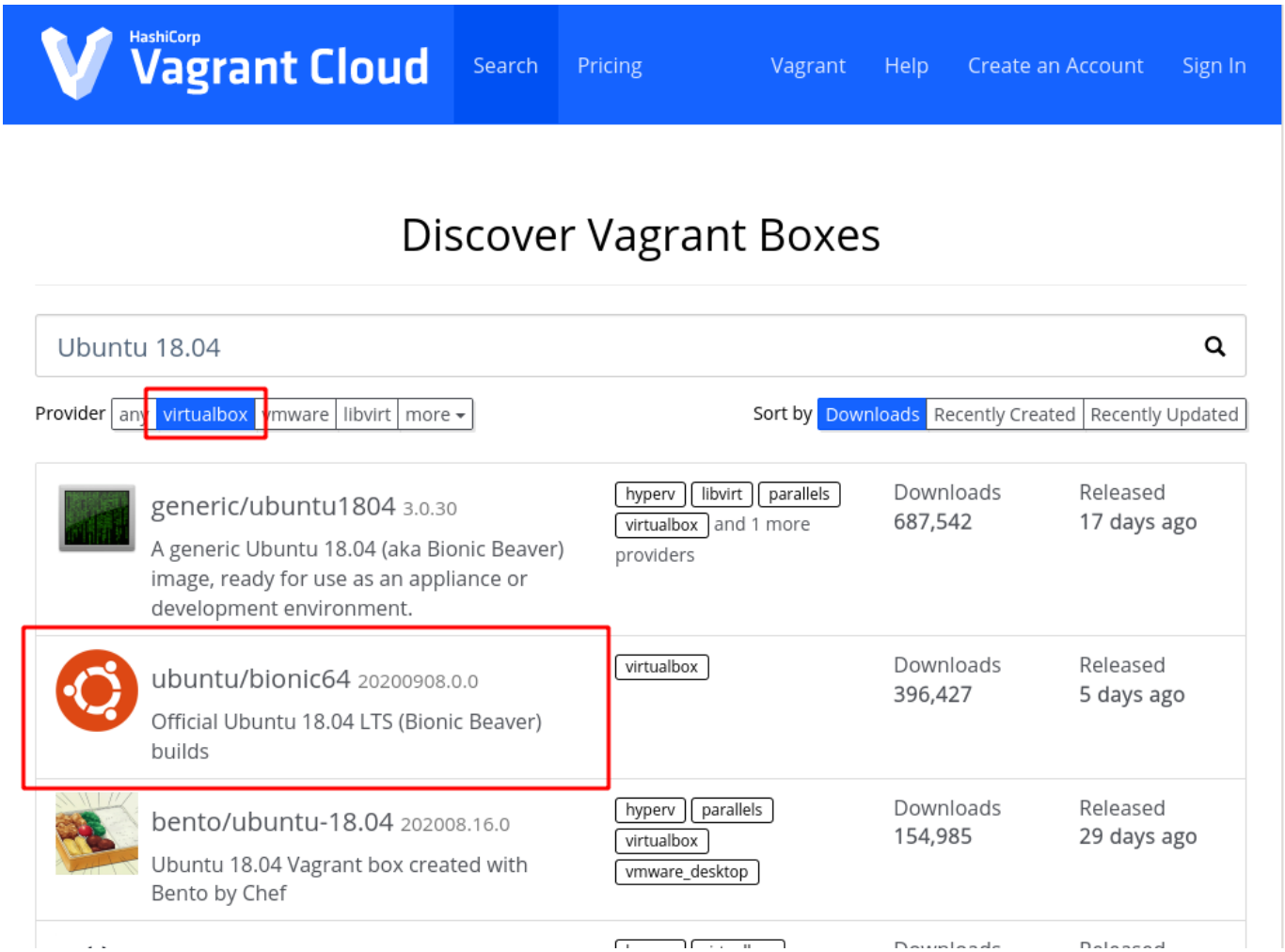
Segue abaixo uma imagem do conteúdo original do arquivo `Vagrantfile` estudado (aberto no editor `vim`)



```
# -*- 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|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You can search for
  # boxes at https://vagrantcloud.com/search.
  config.vm.box = "base"

  # 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.
  # NOTE: This will enable public access to the opened port
  # config.vm.network "forwarded_port", guest: 80, host: 8080

  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine and only allow access
  # via 127.0.0.1 to disable public access
  # config.vm.network "forwarded_port", guest: 80, host: 8080, host_ip: "127.0.0.1"

  # Create a private network, which allows host-only access to the machine
  # using a specific IP.
  # config.vm.network "private_network", ip: "192.168.33.10"

  # 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. These expose provider-specific options.
  # Example for VirtualBox:
  #
  # 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
NORMAL  ./Vagrantfile                                                    unix  utf-8
```

8) Procurar por uma imagem útil para uso no VagrantCloud:

Link para o VagrantCloud: https://app.vagrantup.com/boxes/search



9) Usando editor de texto, modificar o “box” para ubuntu/bionic64

```
# -*- 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|
  # The most common configuration options are documented and commented belo
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You can search fo
  # boxes at https://vagrantcloud.com/search.
  config.vm.box = "ubuntu/bionic64"

  # 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 belo
  # accessing "localhost:8080" will access port 80 on the guest machine.
  # NOTE: This will enable public access to the opened port
```

10) Após finalizar a edição, execute o comando, dentro da pasta ubuntu18

```
# vagrant up
```

```
gitoso@LOVELACE ~/VirtualBox VMs/ubuntu18 vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'ubuntu/bionic64' could not be found. Attempting to find and install...
    default: Box Provider: virtualbox
    default: Box Version: >= 0
==> default: Loading metadata for box 'ubuntu/bionic64'
    default: URL: https://vagrantcloud.com/ubuntu/bionic64
==> default: Adding box 'ubuntu/bionic64' (v20200908.0.0) for provider: virtualbox
    default: Downloading: https://vagrantcloud.com/ubuntu/boxes/bionic64/versions/20200908.0.0/providers/virtualbox.
Download redirected to host: cloud-images.ubuntu.com
==> default: Successfully added box 'ubuntu/bionic64' (v20200908.0.0) for 'virtualbox'!
==> default: Importing base box 'ubuntu/bionic64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'ubuntu/bionic64' version '20200908.0.0' is up to date...
==> default: Setting the name of the VM: ubuntu18_default_1600025261388_59954
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
    default: Warning: Connection reset. Retrying...
    default: Warning: Remote connection disconnect. Retrying...
    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: 5.2.42
    default: VirtualBox Version: 6.1
==> default: Mounting shared folders...
    default: /vagrant => /home/gitoso/VirtualBox VMs/ubuntu18
```

11) Entre no virtualbox e verifique que sua maquina virtual esta rodando.

12) Vamos logar na maquina agora via SSH

13) Dentro da máquina virtual verifique diversas características da VM

14) Dentro da maquina virtual atualize o sistema e instale o htop

15) Saia da maquina guest VM de volta para o host

16) Verifique o estado da VM perante o vagrant

17) Verificar as boxes instaladas

18) Desligue a maquina virtual de maneira "suave"

19) Destrua / Limpe a VM e a box dentro dela, liberando espaço e recursos

Parte II: VM Windows

Gravação:

- Vídeo: ...
- Asciinema: ...

A seguir, a execução dos passos do laboratório será seguida por *screenshots* mostrando o resultado obtido para cada passo.

20) Crie um diretório “windows2010” em Documentos

21) Crie um arquivo Vagrantfile do zero contendo o código fornecido

22) Crie um segundo arquivo de texto com o nome “rdp.ps1” e coloque no mesmo diretório

23) Verifique o virtualbox novamente, veja a VM Windows rodando

24) Acionar o browser e instalar o "chocolatey"

25) Instale o browser "Firefox"

26) Reconfigurar a memoria para 1G RAM, alterando o arquivo Vagrantfile e testar recarregando essa configuração

27) Destruir a VM

Parte III: "Mini-Cluster"

Gravação:

- Vídeo: ...
- Asciinema: ...

A seguir, a execução dos passos do laboratório será seguida por *screenshots* mostrando o resultado obtido para cada passo.

28) Criar um novo diretório "doisubuntu"

29) Criar uma arquivo Vagrantfile novo contendo 2 maquinas virtuais

30) Acionar os ambientes no vagrant

31) Verificar o status e acessar cada uma das maquinas de maneira independente, e atualizar a maquina vm01, fazer teste de ping entre as maquinas

32) Atualizar o Vagrantfile e trocar o box da vm02 por bionic64. E Instalar apache2 na vm02

33) Testar acessar o servidor web com o browser a partir do Host e a partir do vm01 guest usando “wget” no Red Hat

34) Explorando um pouco dos "plug-ins" do Vagrant