



<b>RC-I 2024-2025</b>	<b>TRABALHO DE LABORATÓRIO</b>	<b>Número:</b>	<b>2</b>
CAMADA DE APLICAÇÃO		<b>Data:</b>	
Aplicações Web e HTTP		<b>Prazo:</b>	

## 1. Introdução

Uma das principais ferramentas usadas para lançar ambientes virtuais de desenvolvimento e teste, como “réplicas” escalada de ambientes de produção reais, é o **Vagrant**. O Vagrant permite criar Infraestruturas (Hosts, Redes) em uma máquina desktop/laptop. O Vagrant actua como um wrapper em torno do software de virtualização, acelerando muitas das tarefas associadas à configuração, desmontagem e partilha de Máquinas Virtuais.

O Vagrant é uma ferramenta inestimável para gerir sandboxes locais porque ele cria ambientes multi-node descartáveis, como sistemas autocontidos, usados para desenvolvimento, teste e ganho de experiência com versões de sistemas operativos, aplicações, ferramentas ou configurações, sistemas em rede, e depois os descarta. Esses sistemas sandbox também podem ser usados para reproduzir conectividade e protocolos de rede, problemas de desempenho ou testar interações cliente-servidor.

O Vagrant utiliza um ficheiro de definição (chamado **Vagrantfile**) para definir computadores (uma ou mais Máquinas Virtuais), rede e armazenamento para um ambiente que pode ser criado e executado com uma ferramenta de virtualização local. Como a definição está contida em um ficheiro externalizado, um ambiente Vagrant é reproduzível. O ambiente inteiro pode ser demolido (até mesmo destruído) e então reconstruído com um único comando (usando esse **Vagrantfile**).

O objectivo das experiências com o Vagrant nesta tarefa de laboratório é aprender como configurar um ambiente multinós (servidores, rede, aplicações) e experimentar serviços da camada de aplicação, como um servidor **Web** e o **Hypertext Transport Protocol (HTTP)**.

## 2. Ambientes Vagrant

No LAB #01 usou-se brevemente o Vagrant em um nível bastante alto, para configurar uma Rede Emulada (Mininet).

Se já tentou criar Máquinas Virtuais usadas para testes por meio de uma *Graphical User Interface* (GUI), sabe que pode ser uma dor, e é um processo muito manual (instalar o Sistema Operativo, os pacotes/aplicações, configurá-los, etc.). Também há uma tendência de deixar máquinas de teste espalhadas no seu computador (mesmo rodando e consumindo recursos preciosos) por um longo tempo sem reconstruí-las (ou mesmo desligá-las). Antes de usar o Vagrant, havia uma resistência natural à criação de ambientes limpos, porque há um custo extra de mão de obra associado a fazer isso acontecer, pois é apenas um processo muito manual por meio de uma GUI.



O Vagrant elimina grande parte desse trabalho extra, já que a maior parte dele é completamente automatizado), então vamos nos aprofundar um pouco mais no Vagrant.

Se executar o comando `vagrant` sem nenhum argumento, obterá a saída de ajuda padrão, que exibe as opções de comando disponíveis:

```
1. ConsoleZ
File Edit View Tabs Help
C:\Users\joaojdacosta\Desktop\RC1Projectos>vagrant
Usage: vagrant [options] <command> [<args>]

-h, --help                                Print this help.

Common commands:
  autocomplete    manages autocomplete installation on host
  box             manages boxes: installation, removal, etc.
  cloud           manages everything related to Vagrant Cloud
  destroy         stops and deletes all traces of the vagrant machine
  global-status   outputs status Vagrant environments for this user
  halt            stops the vagrant machine
  help            shows the help for a subcommand
  init            initializes a new Vagrant environment by creating a Vagrantfile
  login           packages a running vagrant environment into a box
  package         manages plugins: install, uninstall, update, etc.
  plugin          displays information about guest port mappings
  port            connects to machine via powershell remoting
  powershell      provisions the vagrant machine
  provision       deploys code in this environment to a configured destination
  push            connects to machine via RDP
  rdp              reloads vagrant machine, loads new Vagrantfile configuration
  resume          resume a suspended vagrant machine
  serve           start Vagrant server
  snapshot        manages snapshots: saving, restoring, etc.
  ssh              connects to machine via SSH
  ssh-config      outputs OpenSSH valid configuration to connect to the machine
  status          outputs status of the vagrant machine
  suspend         suspends the machine
  up               starts and provisions the vagrant environment
  upload          upload to machine via communicator
  validate        validates the Vagrantfile
  version         prints current and latest Vagrant version
  winrm           executes commands on a machine via WinRM
  winrm-config    outputs WinRM configuration to connect to the machine

For help on any individual command run `vagrant COMMAND -h`

Additional subcommands are available, but are either more advanced
or not commonly used. To see all subcommands, run the command
`vagrant list-commands`.
--[no-]color                           Enable or disable color output

Ready
```

As principais opções de comando do Vagrant são utilizadas para gerir o ciclo de vida das Vagrant boxes. Box, ou boxes, é um termo do Vagrant para imagens de “template” de máquina virtual. As Vagrant boxes são “máquinas virtuais especiais”, pois elas normalmente têm vários pacotes necessários já instalados e um **utilizador vagrant** adicionado dentro da imagem. A opção `vagrant box` permite listar, adicionar e remover boxes que o Vagrant conhece:



### 1. ConsoleZ

```
File Edit View Tabs Help
[Icons] Search... Zz ab
C:\Users\joaojdacosta\Desktop\RC1Projectos>vagrant box
Usage: vagrant box <subcommand> [<args>]

Available subcommands:
  add
  list
  outdated
  prune
  remove
  repackage
  update

For help on any individual subcommand run `vagrant box <subcommand> -h`
  --[no-]color          Enable or disable color output
  --machine-readable    Enable machine readable output
-v, --version          Display Vagrant version
  --debug               Enable debug output
  --timestamp           Enable timestamps on log output
  --debug-timestamp     Enable debug output with timestamps
  --no-tty              Enable non-interactive output

C:\Users\joaojdacosta\Desktop\RC1Projectos>
```

A opção `vagrant init` permite inicializar um novo ambiente Vagrant (sem personalizações específicas). Um ambiente Vagrant pode ser uma única Máquina Virtual Vagrant ou uma coleção de Máquinas Virtuais. Então, um ambiente descreverá quais boxes ou Máquinas Virtuais inicializar, junto com todas as configurações associadas, por meio de um ficheiro de configuração com o nome de `Vagrantfile`.

As outras opções de comando comuns são `vagrant up` para inicializar o ambiente, `vagrant halt` para desligar o ambiente, `vagrant global-status` para verificar o status das máquinas virtuais instanciadas, `vagrant destroy` para destruir completamente cada "instância" de uma máquina virtual e `vagrant ssh` para estabelecer uma sessão com cada máquina virtual que está instanciada e em execução.

## 3. Multi-nó com Vagrant

Para cada ambiente Vagrant, ou conjunto de máquinas para experiência, é sempre bom criar um directório de projecto específico no seu sistema, como fez no LAB anterior.

Para iniciar este novo projecto, crie um directório e nomeie-o, por exemplo, `multinode`. Na pasta `multinode` coloque uma cópia do `Vagrantfile` que foi usado no LAB#01 para a experiência Mininet.



Para garantir actualização e aprimoramento adequados para ambientes Vagrant usando o provedor Virtualbox, e caso ainda não tenha feito isso, adicione o plugin `vagrant-vbguest` com o comando:

```
$ vagrant plugin install vagrant-vbguest
```

Ou , se já está instalado

```
$ vagrant plugin update vagrant-vbguest
```

The screenshot shows a terminal window titled "1. ConsoleZ". The window has a toolbar with icons for file operations, search, and help. The main area displays the command output:

```
C:\Users\joaojdacosta\Desktop\RC1Projectos\multinode>vagrant plugin install vagrant-vbguest
Installing the 'vagrant-vbguest' plugin. This can take a few minutes...
Fetching micromachine-3.0.0.gem
Fetching vagrant-vbguest-0.32.0.gem
Installed the plugin 'vagrant-vbguest (0.32.0)'!
```

The status bar at the bottom indicates "Ready", "11720", and "105x8".

### 3.1. Primeiro passo

Utilizaremos uma nova box com um sistema operativo Ubuntu, chamada “ubuntu/trusty64”, para todas as máquinas virtuais nesta experiência.

Abra o `Vagrantfile` com um editor e simplifique-o conforme ilustrado abaixo (removendo também as linhas de comentários, excepto as linhas 1 e 2 (directivas Ruby) antes do `Vagrant.configure("2") do |config|`).

The screenshot shows a code editor with a dark theme. The file is named "Vagrantfile". The code content is as follows:

```
1 # -*- mode: ruby -*-
2 # vi: set ft=ruby :
3
4 Vagrant.configure("2") do |config|
5   config.vm.box = "ktr/mininet"
6
7 end
8
```



O objectivo principal é criar um bloco de código para cada máquina, contendo os comandos de provisionamento específicos. A primeira máquina será chamada de “**webserver**”, como está, abaixo, no bloco de código que começa com config.vm.define "webserver" do |web\_config|. O endereço IP da máquina é um exemplo que pode ser usado normalmente, mas pode precisar adaptar o valor ao endereço de rede privada do Virtualbox em seu sistema.

```
Vagrantfile *  
1 # -*- mode: ruby -*-  
2 # vi: set ft=ruby :  
3  
4 # RCI, 2024. Docente: joaojdacosta@gmail.com  
5  
6 Vagrant.configure("2") do |config|  
7   # config.vm.box = "ktr/mininet"  
8   config.ssh.insert_key = false  
9   config.vbguest.auto_update = false  
10  config.vm.define "webserver" do |web_config|  
11    web_config.vm.box = "ubuntu/trusty64"  
12    web_config.vm.hostname = "webserver"  
13    web_config.vm.network "private_network", ip: "192.168.56.21"  
14    web_config.vm.network "forwarded_port", guest: 80, host: 8080  
15    web_config.vm.provider "virtualbox" do |vb|  
16      vb.name = "webserver"  
17      opts = [ "modifyvm", :id, "--natdnshostresolver1", "on"]  
18      vb.customize opts  
19      vb.memory = "256"  
20    end # do vb  
21    end # do web_config  
22  end # do config  
23 |
```

Agora guarda as alterações realizadas e inicializa o servidor “**webserver**” com o comando:

```
$ vagrant up
```

Observará que o Vagrant começará o download do Ubuntu e, quando terminar, instanciará uma nova Máquina Virtual.



```
C:\Users\joaojdacosta\Desktop\RC1Projectos\multinode>vagrant up
Bringing machine 'webserver' up with 'virtualbox' provider...
==> webserver: Box 'ubuntu/trusty64' could not be found. Attempting to find and install...
    webserver: Box Provider: virtualbox
    webserver: Box Version: >= 0
==> webserver: Loading metadata for box 'ubuntu/trusty64'
    webserver: URL: https://vagrantcloud.com/api/v2/vagrant/ubuntu/trusty64
==> webserver: Adding box 'ubuntu/trusty64' (v20191107.0.0) for provider: virtualbox
    webserver: Downloading: https://vagrantcloud.com/ubuntu/boxes/trusty64/versions/20191107.0.0/providers/virtualbox/unknown/vagrant.box
Download redirected to host: cloud-images.ubuntu.com
Progress: 17% (Rate: 490k*/s, Estimated time remaining: 0:12:01)
```

O processo de download da box pode ser demorado, dependendo da sua conexão a internet... aproveite para tomar um café.

```
webserver: prevent things such as shared folders from working properly. If you see
webserver: shared folder errors, please make sure the guest additions within the
webserver: virtual machine match the version of VirtualBox you have installed on
webserver: your host and reload your VM.
webserver:
webserver: Guest Additions Version: 4.3.40
webserver: VirtualBox Version: 7.0
==> webserver: Setting hostname...
==> webserver: Configuring and enabling network interfaces...
==> webserver: Mounting shared folders...
    webserver: /vagrant => C:/Users/joaojdacosta/Desktop/RC1Projectos/multinode

C:\Users\joaojdacosta\Desktop\RC1Projectos\multinode>
```

Quando estiver pronto, estabeleça uma sessão com o sistema “**webserver**” usando o seguinte comando:

```
$ vagrant ssh
```

A sessão é estabelecida e receberemos o prompt da máquina, semelhante a:



1. ConsoleZ - vagrant@webserver: ~

```
File Edit View Tabs Help
UA Infrastructure Extended Security Maintenance (ESM) is not enabled.

0 updates can be installed immediately.
0 of these updates are security updates.

Enable UA Infrastructure ESM to receive 64 additional security updates.
See https://ubuntu.com/advantage or run: sudo ua status

New release '16.04.7 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

vagrant@webserver:~$
```

Ready

Agora, precisa descobrir qual é o endereço IP da sua máquina abrindo uma nova janela de terminal e executando `ifconfig | grep~inet` para mac/linux ou `ipconfig /all` para windows (ou observe o ícone de propriedades de rede).

Abra a linha de comandos ou terminal do seu sistema e teste a conectividade com o servidor usando o comando `ping`:

```
C:\Windows\system32\cmd.exe
C:\Users\joaojdacosta>ping 192.168.56.21

Pinging 192.168.56.21 with 32 bytes of data:
Reply from 192.168.56.21: bytes=32 time<1ms TTL=64
Reply from 192.168.56.21: bytes=32 time=1ms TTL=64
Reply from 192.168.56.21: bytes=32 time=1ms TTL=64
Reply from 192.168.56.21: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.56.21:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Users\joaojdacosta>
```

Agora, precisa descobrir qual é o endereço IP da sua máquina, para tal na janela de terminal executa `ifconfig | grep~inet` para mac/linux ou `ipconfig /all` para windows (ou observe o ícone de propriedades de rede).

Na janela Terminal do “webserver”, teste a conectividade com seu sistema host da seguinte forma (use seu próprio endereço IP do host, não o do exemplo):



### 1. ConsoleZ - vagrant@webserver: ~

```
File Edit View Tabs Help
[Icons] Search... [Icons]
vagrant@webserver:~$ ping 192.168.8.102
PING 192.168.8.102 (192.168.8.102) 56(84) bytes of data.
64 bytes from 192.168.8.102: icmp_seq=1 ttl=127 time=2.07 ms
64 bytes from 192.168.8.102: icmp_seq=2 ttl=127 time=1.78 ms
64 bytes from 192.168.8.102: icmp_seq=3 ttl=127 time=1.82 ms
64 bytes from 192.168.8.102: icmp_seq=4 ttl=127 time=2.55 ms
64 bytes from 192.168.8.102: icmp_seq=5 ttl=127 time=1.19 ms
64 bytes from 192.168.8.102: icmp_seq=6 ttl=127 time=2.27 ms
64 bytes from 192.168.8.102: icmp_seq=7 ttl=127 time=2.23 ms
64 bytes from 192.168.8.102: icmp_seq=8 ttl=127 time=1.87 ms
64 bytes from 192.168.8.102: icmp_seq=9 ttl=127 time=1.14 ms
64 bytes from 192.168.8.102: icmp_seq=10 ttl=127 time=1.81 ms
^C
--- 192.168.8.102 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9061ms
rtt min/avg/max/mdev = 1.144/1.876/2.558/0.427 ms
vagrant@webserver:~$ █
```

Ready

Agora pode desligar o servidor e verificar se já não está a executar:

```
1. ConsoleZ
File Edit View Tabs Help
[Icons] Search... [Icons]
vagrant@webserver:~$ exit
logout

C:\Users\joaojdacosta\Desktop\RC1Projectos\multinode>vagrant halt
==> webserver: Attempting graceful shutdown of VM...

C:\Users\joaojdacosta\Desktop\RC1Projectos\multinode>vagrant status webserver
Current machine states:

webserver          poweroff (virtualbox)

The VM is powered off. To restart the VM, simply run `vagrant up`'
C:\Users\joaojdacosta\Desktop\RC1Projectos\multinode>█
```

Ready

117

## 3.2. Segundo passo

Edita novamente o Vagrantfile para criar uma réplica do bloco que define o servidor “**webserver**”, e altere naquele bloco de código todas as instâncias de “**webserver**” para “**client**”.



Altere também o endereço IP para um valor diferente, mas na mesma sub-rede, por exemplo 192.168.56.11. Para este servidor, remova a linha que encaminha a porta 80.

Agora inicialize os sistemas “**webserver**” e “**client**” com o comando:

```
$ vagrant up
```

Observará que o sistema “**webserver**” inicializará rapidamente, mas o sistema “**cliente**” levará um pouco mais de tempo, pois precisa ser instanciado como uma nova máquina virtual.

The screenshot shows a terminal window titled "1. ConsoleZ". The window has a menu bar with File, Edit, View, Tabs, Help. Below the menu is a toolbar with icons for file operations like Open, Save, Print, and a search bar labeled "Search...". The main area of the window displays the output of the "vagrant up" command for a "client" VM. The output text is as follows:

```
==> client: Fixed port collision for 22 => 2222. Now on port 2200.  
==> client: Clearing any previously set network interfaces...  
==> client: Preparing network interfaces based on configuration...  
  client: Adapter 1: nat  
  client: Adapter 2: hostonly  
==> client: Forwarding ports...  
  client: 22 (guest) => 2200 (host) (adapter 1)  
==> client: Running 'pre-boot' VM customizations...  
==> client: Booting VM...  
==> client: Waiting for machine to boot. This may take a few minutes...  
  client: SSH address: 127.0.0.1:2200  
  client: SSH username: vagrant  
  client: SSH auth method: private key
```

At the bottom of the window, there is a status bar with the word "Ready".

Quando estiver pronto, estabeleça sessões com os sistemas “**webserver**” e “**client**” usando os seguintes comandos (pode abrir uma janela de terminal para cada um) e verificar os endereços das interfaces, bem como se os servidores podem se comunicar:



1. ConsoleZ - vagrant@webserver: ~

File Edit View Tabs Help

C:\Users\joaojdacosta\Desktop\RC1Projectos\multinode>vagrant ssh webserver

```
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic x86_64)

 * Documentation: https://help.ubuntu.com/

 System information disabled due to load higher than 1.0

UA Infrastructure Extended Security Maintenance (ESM) is not enabled.

0 updates can be installed immediately.
0 of these updates are security updates.

Enable UA Infrastructure ESM to receive 64 additional security updates.
See https://ubuntu.com/advantage or run: sudo ua status

New release '16.04.7 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Oct 17 01:51:43 2024 from 10.0.2.2
vagrant@webserver:~$
```

Ready

A testar a conectividade com o cliente:

1. ConsoleZ - vagrant@webserver: ~

File Edit View Tabs Help

vagrant@webserver:~\$ ping 192.168.56.11

```
PING 192.168.56.11 (192.168.56.11) 56(84) bytes of data.
64 bytes from 192.168.56.11: icmp_seq=1 ttl=64 time=1.53 ms
64 bytes from 192.168.56.11: icmp_seq=2 ttl=64 time=1.91 ms
64 bytes from 192.168.56.11: icmp_seq=3 ttl=64 time=3.32 ms
64 bytes from 192.168.56.11: icmp_seq=4 ttl=64 time=3.79 ms
^C
--- 192.168.56.11 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3024ms
rtt min/avg/max/mdev = 1.536/2.641/3.794/0.941 ms
vagrant@webserver:~$
```

Ready

Consultar os endereços IP:



1. ConsoleZ - vagrant@webserver: ~

```
File Edit View Tabs Help
[Icons] Search... [Icons]
vagrant@webserver:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:5f:bb:e6 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe5f:bb:e6/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:49:26:6a brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.21/24 brd 192.168.56.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe49:266a/64 scope link
        valid_lft forever preferred_lft forever
vagrant@webserver:~$
```

Ready 11720 105x20 105x500 100%

Acessando o terminal do cliente a partir de outro prompt:

2. ConsoleZ - vagrant@client: ~

```
File Edit View Tabs Help
[Icons] Search... [Icons]
1. ConsoleZ - vagrant@... 2. ConsoleZ - vagrant@... +
```

C:\Users\joaojdacosta\Desktop\RC1Projectos>cd multinode

C:\Users\joaojdacosta\Desktop\RC1Projectos\multinode>vagrant ssh client

Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic x86\_64)

\* Documentation: <https://help.ubuntu.com/>

System information disabled due to load higher than 1.0

UA Infrastructure Extended Security Maintenance (ESM) is not enabled.

0 updates can be installed immediately.

0 of these updates are security updates.

Enable UA Infrastructure ESM to receive 64 additional security updates.  
See <https://ubuntu.com/advantage> or run: sudo ua status

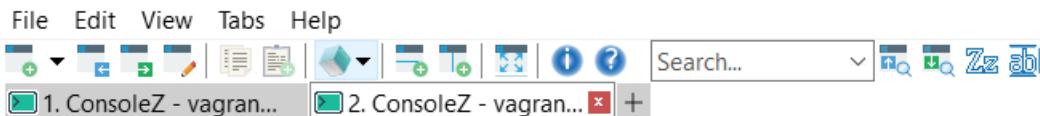
New release '16.04.7 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.

vagrant@client:~\$

Testando a conectividade com o servidor:



### 2. ConsoleZ - vagrant@client: ~



Ready

Notou que tem uma interface de loopback mais duas interfaces de rede (eth0 e eth1) com esquemas de endereçamento IP diferentes?

Agora pode desligar os sistemas “webserver” e “client” e verificar se eles não estão em execução, com os seguintes comandos:

### 2. ConsoleZ



```
C:\Users\joaojdacosta\Desktop\RC1Projectos\multinode>vagrant status
Current machine states:
```

```
webserver          poweroff (virtualbox)
client            poweroff (virtualbox)
```

```
This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.
```

```
C:\Users\joaojdacosta\Desktop\RC1Projectos\multinode>
```

Ready

### 3.3. Terceiro passo

Faça download do ficheiro RCI2024-Lab2-support\_files.zip e descompacte o conteúdo para a pasta do projecto multinode. Um ficheiro chamado bootstrap\_web.sh e uma pasta chamada html serão criados (a pasta html contém um ficheiro chamado index.html).



Edite o Vagrantfile para inserir uma instrução “shell provision” no bloco de código do sistema “webserver”, conforme ilustrado:

```
Vagrantfile
10 config.vm.define "webserver" do |web_config|
11   web_config.vm.box = "ubuntu/trusty64"
12   web_config.vm.hostname = "webserver"
13   web_config.vm.network "private_network", ip: "192.168.56.21"
14   web_config.vm.network "forwarded_port", guest: 80, host: 8080
15   web_config.vm.provider "virtualbox" do |vb|
16     vb.name = "webserver"
17     opts = ["modifyvm", :id, "--natdnshostresolver1", "on"]
18     vb.customize opts
19     vb.memory = "256"
20   end # do vb
21   web_config.vm.provision "shell", path: "bootstrap_web.sh"
22 end # do web_config
23 config.vm.define "client" do |web_config|
```

Esta instrução carregará e executará o script bootstrap\_web.sh dentro da máquina convidada (guest). O script instalará a aplicação do servidor web “apache” (<http://httpd.apache.org>) na máquina e o inicializará.

Para executar o provisionamento e, posteriormente, verificar os resultados, faça o seguinte:

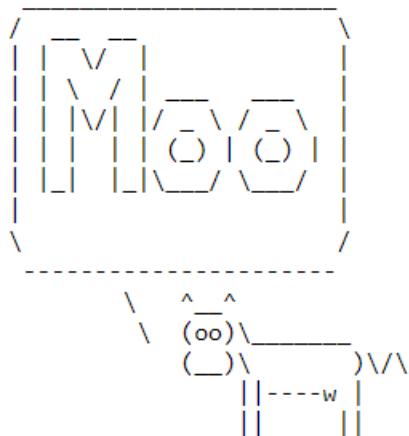
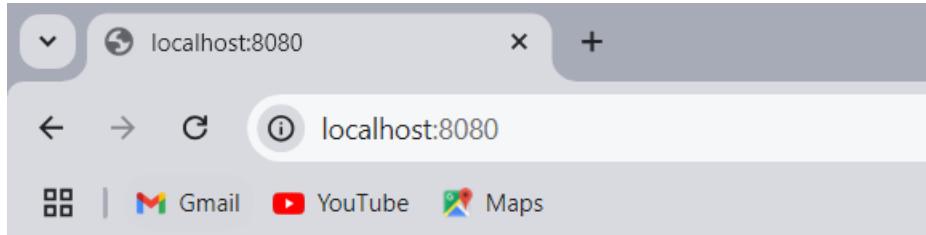
```
$ vagrant up --provision webserver
```

Começará a ver a máquina a inicializar e a ser provisionada (começando com a actualização de pacotes) com o servidor web “apache”.

```
1. ConsoleZ - vagrant up --provision webserver
File Edit View Tabs Help
File Edit View Tabs Help
[...]
webserver: Get:4 https://esm.ubuntu.com trusty-infra-security/main amd64 Packages
webserver: Get:5 http://security.ubuntu.com trusty-security InRelease [56.5 kB]
webserver: Get:6 http://archive.ubuntu.com trusty-backports InRelease [56.5 kB]
webserver: Hit http://archive.ubuntu.com trusty Release.gpg
webserver: Get:7 http://archive.ubuntu.com trusty-updates/main Sources [667 kB]
webserver: Get:8 https://esm.ubuntu.com trusty-infra-security/main Translation-en_US
webserver: Get:9 https://esm.ubuntu.com trusty-infra-updates/main Translation-en_US
webserver: Get:10 http://archive.ubuntu.com trusty-updates/restricted Sources [9,490 B]
webserver: Get:11 http://archive.ubuntu.com trusty-updates/universe Sources [356 kB]
webserver: Get:12 http://archive.ubuntu.com trusty-updates/multiverse Sources [10.2 kB]
webserver: Get:13 http://archive.ubuntu.com trusty-updates/main amd64 Packages [1,172 kB]
webserver: Ign https://esm.ubuntu.com trusty-infra-security/main Translation-en_US
webserver: Ign https://esm.ubuntu.com trusty-infra-security/main Translation-en_US
webserver: Ign https://esm.ubuntu.com trusty-infra-updates/main Translation-en_US
webserver: Ign https://esm.ubuntu.com trusty-infra-updates/main Translation-en_US
```



Quando terminar, abra a seguinte URL: <http://localhost:8080> em um navegador da sua máquina host. Se tudo tiver corrido bem, será recebido com uma mensagem engraçada.



Agora pode encerrar o sistema “**webserver**” com o seguinte comando:

```
$ vagrant halt webserver
```

## 4. Experiência HTTP

Agora que tem a configuração para as duas máquinas e já instanciou ambas, é hora de fazer algumas experiências com o Servidor Web e o HTTP.

### 4.1. Inicializar e conectar as máquinas

Abra uma janela do Terminal e vá para a pasta do projecto multinode. De lá, execute o comando:

```
$ vagrant up
```

Confirme se tem o “client” e o “webserver” em execução usando o comando:

```
$ vagrant global-status.
```

Para acessar a máquina “**client**”, abra uma nova janela do Terminal e usando o comando `vagrant ssh` para esse sistema obterá algo semelhante ao seguinte:



1. ConsoleZ - vagrant@client: ~

File Edit View Tabs Help

C:\Users\joaojdacosta\Desktop\RC1Projectos\multinode>vagrant ssh client

Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic x86\_64)

\* Documentation: <https://help.ubuntu.com/>

System information disabled due to load higher than 1.0

UA Infrastructure Extended Security Maintenance (ESM) is not enabled.

0 updates can be installed immediately.

0 of these updates are security updates.

Enable UA Infrastructure ESM to receive 64 additional security updates.  
See <https://ubuntu.com/advantage> or run: sudo ua status

New release '16.04.7 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Oct 17 02:57:31 2024 from 10.0.2.2  
vagrant@client:~\$ █

Ready

Agora utilizará a aplicação telnet para se conectar aos processos em execução nesses sistemas.

Tente a seguinte interação com o “webserver”:

1. ConsoleZ - vagrant@client: ~

File Edit View Tabs Help

vagrant@client:~\$ telnet webserver 80

telnet: could not resolve webserver/80: Name or service not known

vagrant@client:~\$ █

Ready

**Recebe um erro, certo?** A partir do erro, observará que algo está faltando. Essa parte faltante é o DNS, ou seja, o Domain Name Service para “traduzir” o nome do sistema para seu endereço IP. Então, agora, o que podemos fazer? Veja a próxima seção...

## 4.2. Utilização de alternativa ao DNS

Como não implementamos um servidor DNS, utilizaremos o ficheiro de sistema especial **“/etc/hosts”**, que é utilizado em sistemas nix\* (Linux/Unix) como um tradutor estático de nomes de host para endereços IP. O comando para preencher o ficheiro “/etc/hosts” é o seguinte, e pode



confirmar com o comando “cat” que o ficheiro agora tem uma tradução para o endereço IP do nome “webserver”:

```
vagrant@client:~$ sudo sh -c "echo '192.168.56.21 webserver' >> /etc/hosts"
vagrant@client:~$ cat /etc/hosts
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
127.0.1.1 client client
192.168.56.21 webserver
vagrant@client:~$
```

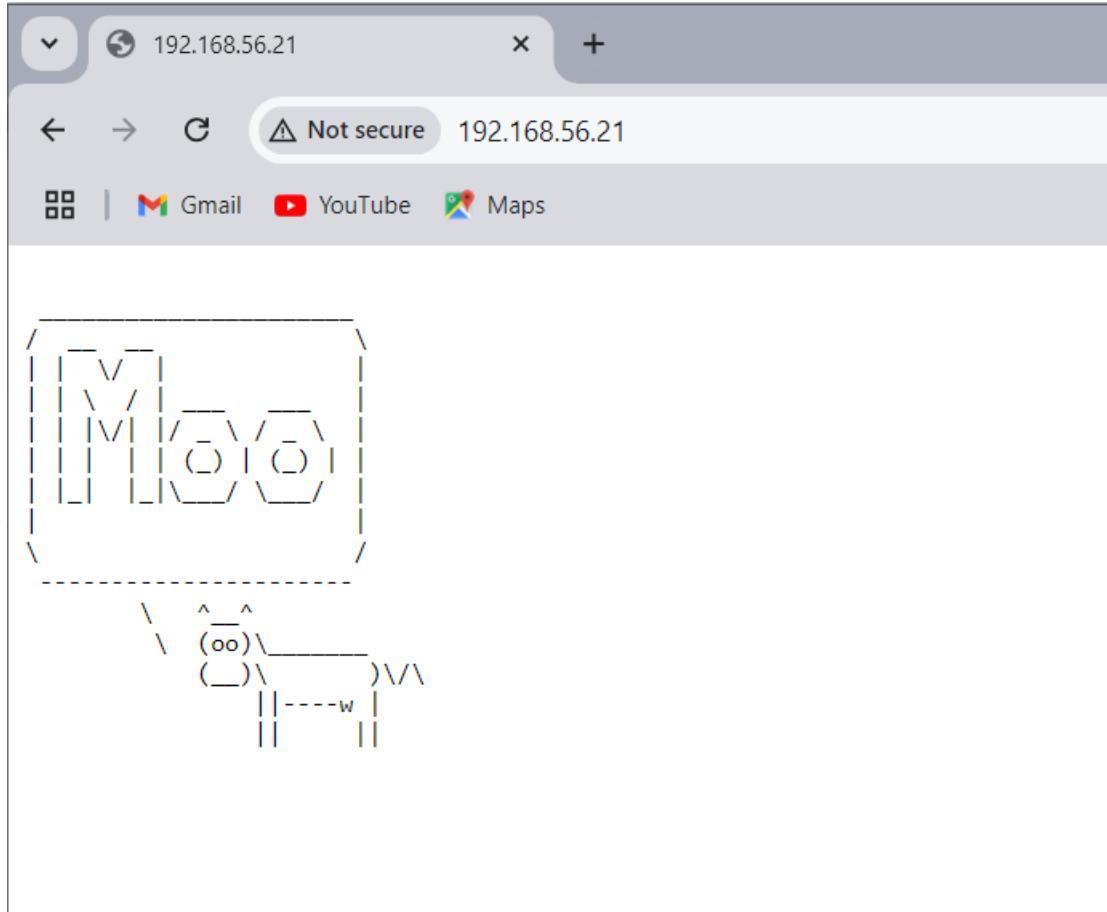
Agora não deve ter problemas com a interação e deverá obter uma página HTML engraçada com esse comando.

```
vagrant@client:~$ telnet webserver 80
Trying 192.168.56.21...
Connected to webserver.
Escape character is '^]'.
```

### 4.3. Conexão com o servidor web pelo browser

Com um navegador em sua máquina host, pode visualizar esse arquivo HTML simples usando o endereço IP da máquina do “webserver” e a porta onde o serviço web está escutando:

<http://192.168.56.21:80>



## 5. Finalização da experiência

Para parar as Máquinas Virtuais e verificar o estado global de todos os ambientes Vagrant activos no sistema, podemos emitir os seguintes comandos:

```
$ vagrant halt  
$ vagrant global-status
```

Confirme se o status das VMs é 'powered off'. Para evitar que essas máquinas instanciadas usem recursos. Depois de apresentar ao docente e escrever o seu memorando, pode destruí-las, pois podem ser recriadas com o comando simples vagrant up. Confirme se não há VMs listadas.

```
$ vagrant destroy  
$ vagrant global-status
```

## 6. Envio dos resultados das experiências

As experiências que executar neste LAB produzirão resultados que precisa relatar ou dos quais será questionado sobre a execução. Para relatar os resultados que alcançou, proceda da seguinte forma:



## 6.1. Procedimento geral

Devia enviar via email um memorando contendo as respostas das questões a serem disponibilizadas e todo o material requerido para aferir a execução do laboratório (screen-shots, saída de linha de comando, código desenvolvido, etc.).

## 6.2. Procedimento específico

Para esta tarefa de LAB, guardará a saída do comando `vagrant up` em um ficheiro chamado “`vagrant.out`”, depois que suas alterações forem feitas no `Vagrantfile`, conforme explicado na Seção 3.3 e, claro, depois de garantir que todo o experimento foi bem executado.

Uma maneira de capturar a saída de um comando pode ser feita conforme mostrado abaixo:

```
$ vagrant halt  
$ vagrant up > vagrant.out
```

Não deve ver nenhuma saída, pois está a ser redirecionado para um ficheiro. No entanto, quando o comando terminar, o cursor retornará para um novo prompt. O ficheiro `vagrant.out` criado pode ser aberto por qualquer editor de texto.

Resumo:

1. Crie um repositório GitHub privado com o nome `rci-labs`. Neste repositório terá a pasta `rci-lab2` onde deve conter os seguintes ficheiros:
  - a. O ficheiro “**vagrant.out**”, que deve ser produzido como explicado acima.
  - b. O ficheiro `Vagrantfile` produzido como explicado na secção 3.3
2. Escreva o memorando com a resposta do questionário e a descrição técnica... deve incluir captura de tela (do telnet a interagir com o “webserver”, etc...)

**Bom trabalho!**