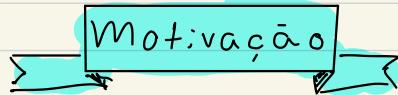


Vagrant | gerenciando máquinas virtuais.



Vagrant



Création de ambientes diversos para
vários projetos. ↳ separados e isolados

Virtualização

- ↳ Rodar um sistema em cima do outro, sem "sujar" a origem ou precisar de outros pc's.

Hypervisor

- ↳ intermediador, que permite a virt.

Vagrant

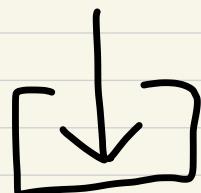
- ↳ Controla o hyper-v.
- ↳ Infraestrutura como código!

Instalação e Primeira VM

Link: vagrantup.com/downloads

1. Instalar VirtualBox primeiro

↳ Como ele atua em cima.



2. Instalar Vagrant

3. Reiniciar o computador

Após instalado, o Vagrant é acessado pelo terminal

Vagrant version → mostra a versão

vagrant init hashicorp/precise64 → inicializa



↳ empresa ↳ nome da box vagrant

↳ pesquisar a mais recente

Vagrant file → guarda configurações do vagrant

box → informações necessárias para rodar as VM's no provider escolhido

Vagrant up → sobe a máquina (precisa de internet)

↳ isso cria a pasta .vagrant

Conexão via SSH ~CFD~

Vagrant status

↳ mostra informações atuais das VM's criadas

Para se conectar remotamente, precisa usar ssh

Vagrant ssh

→ Caso não funcione

↳ Recursos opcionais do windows e
adicional o ssh. (1)

↳ Instalar o Git Bash (1)

↳ Usar PuTTY (3) → puttygen → gerar key

vagrant ssh-config

↳ Mostra informações do ssh

Tipos de Hypervisors

Hypervisor ou monitor de máquina virtual

↳ cria e executa as máquinas

↳ Permite que o computador crie diversas máquinas

Alguns exemplos: Hyper-V, vSphere, Parallels, VMWare
VirtualBox entre outros.

Tipos: Tipo 1: bare metal -> executados no HW host

Tipo 2: rodam como um app no S.O.

Port Forwarding

Acessar algum serviço na máquina usando uma porta específica.

Linha de código:

- ↳ config.vm.network "forwarded-port", guest: 80, host:8080
 - ↳ Configura redirecionamento na porta 80 para 8080 da máquina local

IP na rede privada → IP estático

- ° Quando se quer definir um IP para a máquina

config.vm.network "private-network", ip: "<ip>"

- ↳ Define um IP para a máquina
- ° Isso vai criar um novo adaptador de rede com esse IP

reload
↳ Reinicia

↳ Em modo BRIDGE

- ↳ permite que a máquina participe da sua rede.

DHCP e Destroy

→ Vagrant destroy

↳ Apaga somente a máquina virtual

↳ Não apaga o box nem config

→ DHCP

◦ Para definir dhcp:

config.vm.network "private-network", type: "dhcp"

↳ Isso habilita o servidor DHCP do VB

Public Network → acesso na rede

◦ Ao invés de private, usa "public-network"

↳ Seta como bridge

↳ Só pode colocar estático se for voido



Adicionando chave SSH

Acessando de fora:

ssh vagrant@ip

As vezes pode dar erro
no known-hosts
↳ só apagar o registro
antigo (fingerprint)

Quando conecta com ssh e pede chave:

1. ver onde chave tá → vagrant ssh-config
2. ssh -i path user@ip

Criar a própria chave:

ssh-keygen -t rsa → cria a chave

Existe uma pasta compartilhada do vagrant na
máquina em /vagrant/

- ↳ copiar a chave para a máquina
- ↳ cat <chave> >> .ssh/authorized_keys
- ↳ Agora pode conectar com a chave

Shell Provisioner

↳ Automação -> provisionamento

Shell -> Onde se usa os comandos (executa)

config.vm.provision "shell", (no file)
inline: comando

vagrant provision (CLI)

↳ executa os provisionamentos na máquina

Provisionar significa fornecer tudo que é necessário para rodar um serviço.

Synced Folder

pasta compartilhada

conexão com synced-folder "pasta", "pastaMóvel"

↳ Cria uma pasta compartilhada
(ou tente)

para desabilitar - - ~~, disable: true

Provisioning Shell

- É possível guardar um script em variável

↳ `$variavel = <<-SCRIPT`

`~ ~ \r\nenter`
`~ ~`

`SCRIPT`

Ambiente Mult.-Mach. ne

- Fazer mais de uma máquina acessar um Vagrantfile
 - ↳ Mult.-machine
- Se define um bloco:

```
config.vm.define "nameMag" do |variavel|
  variavel.vm.box = "nameBox"
```

Usando Puppet

↳ Configurar máquina sem usar muito o shell

- Criar pasta **manifests** na pasta compartilhada
- Criar arquivo **.pp**
- Instalar puppet na máquina
- `sudo puppet apply <path do arquivo pp>`

arquivo **PP**:

```
ubuntu20 > configs > manifests > phpweb.pp
1 exec { 'apt-update':
2   command => '/usr/bin/apt-get update',
3 }
4
5 package { [ 'php', 'php-mysql' ]:
6   require => Exec['apt-update'],
7   ensure => installed,
8 }
9
10 exec { 'run-php':
11   require => Package['php'],
12   command => '/usr/bin/php -S localhost:80 &',
13 }
14 |
```

Vagrant file script:

```
1 $script_puppet = <<-SCRIPT
2 sudo apt-get update && \
3 sudo apt-get install -y puppet && \
4 sudo puppet apply /vagrant/configs/manifests/phpweb.pp
5 SCRIPT
```

Integração puppet com vagrant

- Jr no Vagrant file e usar shell provisioner para instalar o puppet.
 - ↳ atualizar e instalar

- Chamar puppet

↳
box.vm.provision "puppet" do |puppet|
 puppet.manifests_path = "caminho"
 || . || file = "caminho.pp"

```
47  
48 |   phpweb.vm.provision "shell", inline: "apt-get update && ap  
49 |  
50 |     # Puppet apply  
51 |     phpweb.vm.provision "puppet" do |puppet|  
52 |       puppet.manifests_path = "./configs/manifests"  
53 |       puppet.manifests_file = "phpweb.pp"  
54 |     end
```

Introdução ao Ansible

→ provisioner

- É a ferramenta mais popular hoje em dia
- Também é executado a partir do host.
- Também é definido em um arquivo mas não precisa de um agent, somente python
- Não executa em windows → precisa de outra máquina linux.
- Instalar através do shell provisioner

```
config.vm.define "ansible" do |ansible|
  ansible.vm.network "public_network", ip: "192.168.15.73"
  ansible.vm.provision "shell",
    inline: "
      apt-get update && \
      apt-get install -y software-properties-common && \
      apt-add-repository --yes --update ppa:ansible/ansible && \
      apt-get install -y ansible
    "
```

- O Ansible usa ssh, então precisa de chaves privadas nele, enquanto a pública fica no destino

```
66
67   ansible.vm.provision "shell",
68     inline: "cp /vagrant/idUbuntu /home/vagrant && \
69       chmod /home/vagrant/idUbuntu 600"
70
```

Vagrant validate
↳ Verifica a integridade do Vt

- Basicamente, o Ansible empurra comandos via SSH para as máquinas virtuais.

Testando Playbook

- Criar pasta no configs para guardar os arquivos
hosts -- define qual máquina será configurada com IP e outras informações

```
1 [mysqlserver]
2 192.168.15.72
3
4 [mysqlserver:vars]
5 ansible_user=vagrant
6 ansible_ssh_private_key_file=/home/vagrant/idUbuntu |
7 ansible_python_interpreter=/usr/bin/python3
8 ansible_ssh_common_args=' -o StrictHostKeyChecking=no'
```

playbook/provision -- passos para instalação

- ↳ A explicação será feita no curso de Ansible para executar o roteiro da máquina do Ansible
- ↳ `ansible-playbook -i caminho host caminho playbook`

Corrigindo erros

- Ansible não queria instalar o mysql e então reinstalei seguindo os passos da documentação.
- ↳através do pip

```
72     ansible.vm.provision "shell",
73         inline: "
74             apt-get update && \
75             python3 -m pip install --user ansible && \
76             python3 -m pip install --user ansible-core==2.12.3 && \
77             python3 -m pip install --upgrade --user ansible
78         "
```

Além disso, altere: a ordem de instalação do mysql, já que precisa do python3-mysqldb antes:

```
10     - name: 'Instalar MySQL Server'
11       apt:
12           update_cache: yes
13           cache_valid_time: 3600 #1 hora
14           name: ["python3-mysqldb", "mysql-server"]
15           state: latest
16           become: yes
```

Integração com Vagrant

- Chamar o ansible pelo VF.

Ansible local → ansible na máquina que está no lado

Ansible provisioner → também precisa na máquina

Shell → chamará ansible-playbook

```
# Chama o ansible na máquina
ansible.vm.provision "shell",
  inline:
    ansible-playbook -i /vagrant/configs/ansible/hosts /vagrant/configs/ansible/playbook.yml
"
```

Puppet x Ansible

- Ansible -> provisionamento
 - ↳ tudo no playbook vira python, por isso precisa ter na máquina.
 - ↳ o PB deve ser executado em cada máquina
- Puppet -> Gerenciamento de configuração
 - ↳ com o arquivo manifest, definimos a configuração da máquina que usará o puppet-agent
 - ↳ self-healing
 - ↳ Verifica config. em um intervalo

Configuração do Provider

Config.rw. provider "provedor" do /pl

p. memory (- memoria) = 1024 (MB)

p. cpus (-o quantas cpus) = 2

↳ Pode mudar algumas configs

Box e Global Status

↳ pode configurar box individual → só adicionar no define da máquina

global-status

↳ mostra todos os ambientes disponíveis no computador

↳ mostra ID que pode ser usado em comandos a partir de qualquer pasta

↳ só faz parte quem já foi criado

-- prune

↳ Mostra só as configurações existentes e remove antigas/não usadas

vagrant d

↳ pasta onde há as boxes

↳ fica no usuário

vagrant box list

↳ mostra os boxes instalados

Virtualização vs Container

Container

- ↳ tem o mesmo objetivo de isolamento de máquinas
- ↳ Faz de forma mais leve
 - ↳ Container engine (Docker)
 - ↳ Só roda o que precisa: descindo no Linux
 - ↳ Usa o Linux como base.

Virtualização

- ↳ Usa hardware pois tem que carregar um SO interno
- ↳ Não aproveita nem huma biblioteca do host