Vagrant: Gerenciando máquinas virtuais

- Esse curso capacita a:
  - Criar máquinas virtuais;
  - Automatizar a criação de uma máquina;
  - Provisionar tarefas integrando Ansible e Puppet;
  - Configurar ambientes de software;
  - Saber como criar e destruir máquinas sem medo;

Aulas:

1. **A primeira máquina virtual:**
   - VirtualBox, VMware, Hyper-V, entre outros, são Hypervisors;
   - Um Hypervisor emula o hardware do computador para criar e executar máquinas virtuais;
   - O Vagrant é uma ferramenta que controla o Hypervisor a partir de um arquivo simples, o Vagrantfile;
   - O Vagrantfile define detalhes da máquina virtual, como o sistema operacional, a rede, software utilizado, etc.;
   - O comando **vagrant init <box>** cria um Vagrantfile;
   - A box é baixada da internet e possui a imagem do sistema operacional, entre outras configurações;
   - Para inicializar e rodar a VM com Vagrant, usa-se o comando: **vagrant up**;
   - O comando **vagrant status** mostra detalhes sobre o status da máquina virtual;
   - Para se conectar com a máquina virtual, usamos a ferramenta SSH;

2. **Configuração da rede:**
   - Existem 3 formas para configurar a rede:
     - Forwarded Port;
     - Private Network;
     - Public Network;
   - Na configuração *Forwarded Port*, mapeamos uma porta do host para o guest, por exemplo:
     config.vm.network "forwarded_port", guest: 80, host: 8080,
   - Na *Private Network* (static ou dhcp) é usado um endereço privado que não é acessível na sua rede pública (por exemplo, a rede empresarial);
   - Na *Public Network* (static ou dhcp), usamos um endereço que faz parte da sua rede pública (por exemplo, da rede empresarial);
   - Com o comando **vagrant halt** podemos parar a execução da máquina virtual;
   - O comando **vagrant reload** recarrega a configuração da máquina virtual;

3. **Lidando com SSH:**
    - O comando **vagrant ssh-config** lista as configurações SSH que o comando **vagrant ssh** usará;
    - O Vagrant gera automaticamente um par de chaves SSH;
    - A chave pública fica na máquina virtual (guest), a chave privada fica no host;
    - No arquivo **.ssh/know_host**, fica guardado o fingerprint de cada máquina com qual o SSH se conectou;
    - Como criamos máquinas através do Vagrant com frequência, é preciso limpar esse arquivo **.ssh/know_host** (ou apagar) de tempos e tempos;
    - Para gerar um par de chaves SSH, existe a ferramenta ssh-keygen;
    - A chave pública deve ficar dentro do arquivo .ssh/authorized_keys da máquina virtual;

4. **Provisionando a máquina:**
    - O provisionador mais simples é o Shell Provisioner;
    - Provisionamento significa instalar e configurar tudo o que for necessário para rodar algum serviço ou aplicação;
    - Para usar o Shell Provisioner, basta definir um script com os passos de instalação:

```
Vagrant.configure("2") do |config|
    config.vm.provision "shell", path: "script.sh"
end
```

    - Os comados do Shell Provisioner também podem ser usados de maneira inline ou remoto:

```
$script = <<-SCRIPT
  echo Instalando MySQL
SCRIPT

Vagrant.configure("2") do |config|
  config.vm.provision "shell", inline: $script
end
-----------------------------------------------
                |    OU    |
-----------------------------------------------
Vagrant.configure("2") do |config|
  config.vm.provision "shell", path: "https://seu-servidor/script.sh"
end
```

    - O Vagrant automaticamente compartilha uma pasta entre o host e o guest (Synced Folder);
    - Por padrão, é compartilhada a pasta onde se está o Vagrantfile;
    - Na máquina guest, podemos acessar a pasta pelo caminho /vagrant;
    - A pasta compartilhada pode ser reconfigurada no Vagrantfile: **config.vm.synced_folder "src/", "/public"**

5. **Conhecendo Puppet:**
    - No mesmo Vagrantfile, podemos configurar várias máquinas, separando as configurações (Multi-Machine);
    - O Puppet é uma ferramenta popular para provisionar uma máquina;
    - Provisionamento significa instalar e configurar tudo o que for necessário para rodar algum serviço ou aplicação;
    - Com Puppet, podemos definir os passos de instalação de mais alto nível, facilitando a manutenção;
    - Os passos de instalação são configurados em um arquivo *manifest*, com a extensão .pp;
    - Para rodar o Puppet, é preciso instalar um cliente na máquina virtual;
    - O Vagrant integra e consegue chamar o Puppet a partir do comando **vagrant provision**;
    - Ao rodar o comando **vagrant up** pela primeira vez, ele também roda o provisionamento;
    - Para configurar o Puppet dentro do Vagrantfile, basta usar:

```
config.vm.provision "puppet" do |puppet|
  puppet.manifests_path = "manifests"
  puppet.manifest_file = "wep.pp"
end
```

6. **Usando Ansible:**
    - O Ansible, assim como o Puppet, é uma ferramenta para provisionar uma máquina;
    - O Ansible envia comandos SSH para a máquina a ser configurada, e não precisa de um cliente instalado (apenas o Python);
    - Os passos de instalação são configurados de alto nível, dentro de um playbook;
    - O arquivo de inventário (hosts) define os alvos da instalação;
    - O Vagrant integra o Ansible e tem uma configuração dedicada:

```
config.vm.provision "ansible" do |ansible|
 ansible.inventory_path = "hosts"
 ansible.playbook = "playbook.yml"
end
```

7. **Configurações do provedor:**
    - No Vagrantfile, podemos definir configurações específicas do provedor (hypervisor);
    - As configurações são referente à memória, CPU, rede ou interface gráfica, entre outras opções;
    - Para listar todos os boxes baixadas, use o comando: **vagrant box list**;
    - Para remover as boxes desatualizadas: **vagrant box prune** ou **vagrant box remove <nome>**;

- Para listar todas as máquinas que foram criadas no host use: **vagrant global-status –prune**;
- Através do ID da máquina, podemos controlar a máquina virtual fora da pasta do projeto, por exemplo:
  vagrant destroy –f <ID-da-VM>;

8. **Container vs Virtualização:**
   - O Docker é uma tecnologia para criar, rodar e administrar containers, baseado no Linux;
   - Containers virtualizam o sistema operacional;
   - Máquinas virtuais virtualizam o hardware;
   - Containers são mais leves do que máquinas virtuais;
   - Ambos, containers e máquinas virtuais, servem para rodar e isolar processos e aplicações;