05

Instalando Ansible e o primeiro comando

Transcrição

Anteriormente, explicamos o que é infraestrutura como código, cultura DevOps e o papel do Ansible. Comentamos como será nosso ambiente de desenvolvimento para o curso inteiro.

Nós vamos usar a máquina de desenvolvimento para simular a máquina de controle, e dentro da máquina de desenvolvimento, **criaremos uma máquina virtual** usando Virtual Box e Vagrant. Esta máquina será configurada para instalar o nosso estudo de caso, para isto precisaremos ter funcionando na nossa máquina Ansible.

No Mac OS X, precisaremos apenas usar o `brew`. Eu já tenho instalado os pacotes que são o Python e o próprio Ansible na minha máquina.

Se você ainda não os tiver, quando abrir o *Install*, eles serão instalados e será desnecessário fazer outra configuração. Depois que rodarmos, provaremos como Python e Ansible estão funcionando.

> Se você for usuário Linux, encontrará orientações para a instalação nos exercícios (https://cursos.alura.com.br/course/infraestrutura-como-codigo-com-ansible/task/34101). O processo é bem semelhante, você terá que usar o comando `apt-get` em vez de `brew`. No mesmo link, disponilizaremos orientações para outros pacotes.

> No entanto, até o momento, de acordo com a documentação (http://docs.ansible.com/), o Ansible não tem suporte para usar o Windows como máquina de controle. Se este é o seu sistema operacional, você deverá criar uma máquina virtual para gerar sua máquina de controle também. Porem, o bash do ubuntu pode ser ativado nas versões de windows 10 posteriores à atualização creators update, possibilitando a instalação e o uso de todos os comandos necessários no curso. O passo a passo pode ser encontrado no blog oficial da Microsoft: https://blogs.msdn.microsoft.com/luisdem/2016/09/01/bash-on-windows-passo-a-passo/ (https://blogs.msdn.microsoft.com/luisdem/2016/09/01/bash-on-windows-passo-a-passo/).

> Você pode baixar o VirtualBox (https://www.virtualbox.org/) para isso, na parte de download encontraremos os binários para cada distribuição. Vagrant está disponível no site da empresa, na seção de downloads (https://www.vagrantup.com/downloads.html). Nela, encontraremos um instalador para cada distribuição.

Receberemos o seguinte retorno no Terminal:

```
Warning: python 2.7.14 is already installed
Warning: ansible 2.4.2.0_1 is already installed
alura$ python -V
Python 2.7.10
alura$ ansible --version
ansible 2.4.2.0
```

Digitaremos o comando `-v` para saber a versão do Python e `--version` para identificarmos a versão do `ansible`.

Para prosseguirmos, precisaremos tanto do VirtualBox, como do Vagrant. Depois da instalação, basta abrir o PKG e seguir as instruções. Como eu realizei o processo na minha máquina, conseguiremos rodar o Vagrant na linha de

comando. Por exemplo, veremos qual é a versão atual.

```
$ vagrant --version
Vagrant 2.0.1
```

Para termos certeza que a distribuição do Ansible está funcionando, vamos rodar o comando Ansible contra uma máquina virtual de exemplo.

```
$ cd wordpress_com_ansible
$ ls
Vangrantfile        provisioning.retry         world.txt
group_vars          provisioning.yml
hosts               roles

$ atom
```

Nós vamos criar a nossa **primeira máquina virtual**, mostraremos como ela pode ser acessada e, depois, rodaremos "Hello, World" na linha de comando. Começaremos criando um arquivo chamado `vagrant.file` , no qual será feita a configuração da máquina virtual. Como está fora do escopo do curso ensinar Vagrant em detalhes, você encontrará um arquivo (https://s3.amazonaws.com/caelum-online-public/746-ansible/01/arquivos/Vagrantfile) para usá-lo como base.

A seguir, dispararemos a máquina virtual. Ele já determina que haverá uma máquina virtual chamada `wordpress` .

```
Vagrant.configure("2") do |config|

    config.vm.box = "ubuntu/trusty64"

    config.vm.provider "virtualbox" do |v|
        v.memory = 1024
    end

    config.vm.define "wordpress" do |m|
        m.vm.network "private_network", ip: "172.17.177.40"
    end

end
```

Vamos iniciar essa máquina virtual?

```
$ vagrant up
Bringing machine 'wordpress' up with 'virtualbox' provider...
==> wordpress: Checking if box 'ubuntu/trusty64' is up to date...
```

A máquina foi criada, mas aguardaremos a importação do snapshot. Talvez, na sua máquina, esse processo deve demorar um pouco mais porque será necessário baixar a imagem do Ubuntu.

Enquanto ele levanta a máquina virtual, criaremos o primeiro arquivo que será usado pelo Ansible para começar a infraestrutura completa do código.

O nome do arquivo será `hosts`, um arquivo de inventário onde colocaremos todas as informações necessárias para o Ansible saber como acessar as máquinas. Em seguida, vamos colocar o IP da máquina criada: `172.17.177.40`. Nós também adicionaremos o grupo `wordpress`, cuja utilidade é informar ao Ansible para o servidor qual é a sua serventia.

Você pode, até o momento, ter a impressão de que ele não é muito útil, porém, veremos como o Ansible decide o que a sua máquina é, além de qual tipo de código será aplicado na estrutura. Salvaremos o arquivo nesse formato, e depois, rodaremos o comando `ssh`.

```
$ vagrant ssh
```

Nós conseguiremos verificar que a máquina existe, após limparmos a tela com o comando `clear`, rodaremos o comando Ansible:

```
$ ansible -u vagrant -i hosts -m shell -a 'echo Hello, World'
```

O Vagrant sempre cria um usuário chamado `vagrant`, além dele, passaremos o arquivo de inventário `hosts` e o módulo `shell`. Módulos são os comandos que o Ansible são capazes de rodar, o primeiro deles é responsável por executar um comando no Shell quando rodarmos o SSH na máquina. Nós queremos passar `echo Hello, World`.

No entanto, ao rodarmos o comando, teremos um erro porque esquecemos de informar para o Ansible contra quais grupos estamos executando determinado comando. No caso, especificaremos o grupo `wordpress`.

Vamos trabalhar com um usuário SSH, então, precisamos passar uma senha ou uma chave. Se disponibilizarmos a senha, não vai funcionar corretamente. Passar uma **chave** é uma melhor solução, faremos isso adicionando `--private-key`. O Vagrant criou uma chave: `.vagrant/machines/wordpress/virtualbox/private_key`

```
$ ansible -u vagrant --private-key .vagrant/machines/wordpress/virtualbox/private_key -i hosts
```

Nós passamos:

- grupo de hosts que queremos rodar o comando;
- qual nome do usuário;
- a chave privada (e evitar passar senha na linha de comando);
- informamos o arquivo de inventário para ele conferir os hosts que configuraremos;
- informamos o módulo que executaremos: shell;
- quais os argumentos que estamos passando: `echo Hello, World` (comando que eu executaria direto no bash).

Agora conseguiremos executar o comando.

```
$ ansible wordpress -u vagrant --private-key .vagrant/machines/wordpress/virtualbox/private_key
172.17.177.40 | SUCCESS | rc=0 >>
Hello, World
```

Esse retorno do Ansible significa que ele rodou um comando no servidor `172.17.177.40`, e que a execução foi bem-sucedida. A saída foi `Hello, World`.

No entanto, é possível compreender se ele fez uma conexão SSH? Adicionaremos o parâmetro `-vvvv` na execução, assim iremos os assegurar. O Ansible vai ficar mais verboso quando for executado, mas conseguiremos ver exatamente o que está acontecendo. Isto é bastante útil quando estamos depurando ou desenvolvendo um script'.

```
$ ansible -vvvv wordpress -u vagrant --private-key .vagrant/machines/wordpress/virtualbox/privat
```

Agora a saída será mais extensa e nos informará qual é a versão e arquivos utilizados, além de como ele achou os módulos no `path` . Ele nos avisará, inclusive, qual usuário foi utilizado para estabelecer a conexão, além do comando SHH usado.

Depois, será mostrado como são carregados os arquivos dos comandos enviados. No fim, ainda teremos a saída simplificada nos informando que a execução foi bem-sucedida.

Se você tiver alguma dúvida sobre o que aconteceu do ponto de vista de conectividade no Ansible, caso tenha algum problema para logar, um usuário em que antes conseguimos acessar a máquina, mas não é mais possível, ou outras informações sobre problemas, o padrão `-vvvv` pode ser útil para ajudar a depurar.

Nossa instalação Ansible está funcionando, conseguimos nos conectar via SSH na máquina virtual criada, sabemos que ela está executando os comandos que o Ansible está pedindo. Estamos prontos para criar o playbook a seguir.