02

Download e Instalação da aplicação

Transcrição

Configuramos o usuário do MySQL e mostramos como configurar um usuário com as permissões necessárias para o Wordpress poder salvar os dados do MySQL, da estrutura de tabelas. Vimos também como navegar na documentação de módulos do Ansible, agora, falta construirmos a instalação do Wordpress. Tanto o PHP quanto o Apache está funcionando, assim como o MySQL.

Primeiramente, precisaremos saber qual é o método de instalação da nossa preferência. Até o momento, usamos pacotes `apt`, mas esta não é a forma preferida de fazer instalação de Wordpress. Por quê? Porque eles dificilmente seguem atualizados com as versões mais recentes do Wordpress, e este é um pré-requisito.

Vale lembrar que o Wordpress tem um software muito ativo, com correções e melhorias constantes, além da forma como se lida com dependências do instalador de um pacote para outro pode mudar de acordo com a distribuição de Linux que estamos usando, isso pode se tornar um problema.

Como realizaremos esta instalação? Faremos o download da versão oficial, disponível no site (https://wordpress.org/download/), descompactaremos e configuraremos - use o arquivo de exemplo disponibilizado na instalação do Wordpress -, incluiremos alguns parâmetros de banco de dados, para que ele se conecte com aquele criado por nós.

Em seguida, vamos configurar o Apache instalado anteriormente, para que seja reconhecido o site do Wordpress. Nossa instalação estará concluída.

Faremos o download usando os módulos do Ansible. Teremos que identificar a URL para realizarmos o download da última versão: wordpress.org/latest.tar.gz (wordpress.org/latest.tar.gz). O próximo passo será consultar novamente a documentação do Ansible, descobriremos que `get_url` poderá ser utilizado na instalação, por fazer downloads de arquivo via HTTP, HTTPS ou FTP.

Ao acessarmos a documentação específica (http://docs.ansible.com/ansible/latest/modules/get_url_module.html), encontraremos na Sinopse, a descrição de como ele se comporta com `_proxy`, com o protocolo HTTP. Encontramos também o aviso de que se você executá-lo em um **host remoto Windows**, será necessário o módulo `win_get_url`. Veremos um exemplo útil, que faz download do arquivo `foo.conf` e será utilizado por nós.

Vamos novamente usar o exemplo como referência, lembrando de fazer ajustes na indentação como fizemos em todos os módulos.

```
- name: Download foo.conf
  get_url:
    url: http://example.com/path/file.conf
    dest: /etc/foo.conf
    mode: 0440
```

Em seguida, editaremos `name` incluindo sua utilidade e a `url` também será alterada.

```
- name: 'Baixa o arquivo de instalacao do Wordpress'
  get_url:
```

```
    url: 'https://wordpress.org/latest.tar.gz'
    dest: '/tmp/wordpress.tar.gz'
    mode: 0440
```

O destino (`dest`) referente para onde vamos copiar, passamos `/tmp/wordpress.tar.gz` . Antes de modificar `mode` , consultaremos novamente a documentação, na qual veremos que este parâmetro é responsável por informar quais permissões serão aplicadas no arquivo que estamos salvando. Temos a opção de configurá-las como já é feito no sistema operacional Linux. Será desnecessário adicionar uma permissão especial para o download, isto significa que podemos remover o `mode` de trecho do código e deixá-lo usar as permissões padrão. Como estamos salvando no `/tmp` , é irrelevante o usuário saber em qual salvaremos.

Na sequência, descompactaremos o arquivo usando o módulo `unarchive` , depois de copiado na máquina local (esta parte é opcional). Podemos usar o exemplo dado na documentação.

```
- name: 'Baixa o arquivo de instalacao do Wordpress'
  get_url:
    url: https://wordpress.org/latest.tar.gz'
    dest: '/tmp/wordpress.tar.gz'

- name: 'Descompacta o wordpress'
  unarchive:
    src: '/tmp/wordpress.tar.gz'
    dest: /var/www/
```

Observe que preenchemos `src` com o mesmo destino usado em `dest` do módulo `Baixa o arquivo de instalacao do Wordpress` . Já o destino do módulo `Descompacta o wordpress` , por padrão, nós instalamos aplicações Web em `var/www/` . No caso, será desnecessário criar o diretório chamado `wordpress` porque ele foi gerado automaticamente.

No entanto, `var/www` é um diretório que apenas `root` tem a permissão para fazer alterações. Como faremos isso no Ansible? Usando a tag `become` , configurada com `yes` . Ela ficará no mesmo nível da task, por não se tratar de um parâmetro.

```
- name: 'Descompacta o wordpress'
  unarchive:
    src: '/tmp/wordpress.tar.gz'
    dest: /var/www/
  become: yes
```

Em seguida, testaremos se o código está funcionando usando o comando `ansible-playbook` no terminal.

```
xxx-iMac:wordpress_com_ansible caelumrio$ ansible-playbook -i hosts provisioning.yml
```

Após o processo de download, veremos que a tarefa de descompactação do Wordpress não vai funcionar. Receberemos a mensagem de que ele não conseguiu acessar `tmp/wordpress.tar.gz` .

```
TASK [Baixa o arquivo de descompactação do Wordpress] ***************************
changed: [172.17.177.40]

TASK [ Descompacta o wordpress] *************************************************
fatal: [172.17.177.40]: FAILED! => {"changed": false, "msg": "Could not find or access '/tmp/ w
        to retry, use: --limit @/Users/marcoscropalato/wordpress_com_anisble/provisioning.retry

PLAY RECAP **********************************************************************
172.17.177.40              : ok=5    changed=1    unreachable=0    failed=1
```

Por que ocorreu essa falha? Precisaremos descobrir, pesquisando na documentação do `unarchive`. Na sinopse do comando, está definido a necessidade de configurar `remote_src=yes` para descompactar um arquivo que já existe no alvo. Veremos mais informações sobre `remove_src` na seção específica dos parâmetros.

Nele, é explicado que quando a configuração for `yes`, ele vai buscar dentro da máquina (do host), pelo arquivo que tentamos descompactar. Caso ele esteja configurado com `no`, a busca será feita na máquina de controle - que tenta se comunicar com o host.

Como não fizemos o download do arquivo para a máquina e, depois, subimos para remoto - passamos diretamente para esta parte -, configuraremos `remote_src=yes`. Faremos isso a seguir:

```
- name: 'Descompacta o wordpress'
  unarchive:
    src: '/tmp/wordpress.tar.gz'
    dest: /var/www/
    remote_src: yes
  become: yes
```

Agora, esperamos rodar o comando no terminal e que o funcionamento seja conforme o esperado. Desta vez, ele não tentará baixar novamente o wordpress e vai ter **sucesso na desinstalação**.

```
TASK [Baixa o arquivo de instalação do Wordpress] *******************************
changed: [172.17.177.40]

TASK [ Descompacta o wordpress] *************************************************
changed: [172.17.177.40]

PLAY RECAP **********************************************************************
172.17.177.40              : ok=6    changed=1    unreachable=0    failed=1
```

Vamos verificar o que aconteceu dentro da máquia virtual:

```
xxx-iMac: wordpress_com_ansible caelumrio$ vagrant ssh
```

Depois, acessaremos `var/www` e encontraremos a pasta `wordpress` criada. Podemos verificar ainda seu conteúdo usando `ls wordpress`:

```
vagrant@vagrant-ubuntu-trusty-64:~$ cd /var/www
vagrant@vagrant-ubuntu-trusty-64:/var/www$ ls -1h
total 8.0K
drwxr-xr-x 2 root   root    4.0K Dec  7 13:52 html
drwxr-xr-x 5 root   root    4.0K Nov 29 19:06 wordpress
vagrant@vagrant-ubuntu-trusty-64:/var/www$ ls wordpress/
index.php     wp-active.php      wp-comments-post.php   wp-cron.php        wp-load.php     wp-se
license.txt   wp-admin           wp-config-sample.php   wp-includes        wp-login.php    wp-s
readme.html   wp-blog-header.php wp-content             wp-links-opml.php  wp-mail.php     wp-t
```

Nós já instalamos o Wordpress, agora, o que nos falta é configurar o Wordpress para que ele localize o banco de dados, saber como ele acessa, e avisar ao Apache onde está o Wordpress e como ele será utilizado. Faremos isso a seguir.