04

Alterando a configuração no Host

Transcrição

Vimos como baixar o Wordpress, como descompactá-lo usando dois módulos novos (`get_url` e `unarchive`). A seguir, daremos continuidade na instalação do Wordpress, falta informá-lo como acessar o banco de dados e avisar para o Apache como ele servirá ao Wordpress.

Começaremos a fazer configuração do Wordpress, para isto, precisaremos de um arquivo de configuração. Acessaremos a máquina virtual:

```
$ vagrant ssh
```

Se usarmos o comando `ls` no diretório `wordpress` , veremos que ele tem um arquivo de exemplo.

```
vagrant@vagrant-ubuntu-trusty-64:~$ cd /var/www/wordpress
vagrant@vagrant-ubuntu-trusty-64:/var/www/wordpress$ ls
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
```

Um dos arquivos que encontraremos na listagem é `wp-config-sample.php` , nós devemos copiá-lo para que ele se torne a configuração de Wordpress e poder ser alterado conforme a nossa necessidade. Como fazemos isso no Ansible? Da lista de módulos (http://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html) do Ansible, precisaremos doCopy (http://docs.ansible.com/ansible/latest/modules/copy_module.html#copy-module), utilizado para copiar arquivos com localizações locais ou remotas para outra que seja remota.

Nos exemplos, primeiramente, vemos como é feita a cópia de um arquivo na máquina local para uma remota. Não se enquadra no que buscamos. De fato, o que buscamos é copiá-lo de uma máquina remota para outra, alterando o nome. Encontraremos um exemplo que se enquadra e será usado como referência no código.

```
- name: Copy a "sudoers" file on the remote machine for editing
  copy:
    src: /etc/sudoers
    dest: /etc/sudoers.edit
    remote_src: yes
    validate: /user/sbin/visudo -cf %s
```

Modificaremos o `src` , no qual adicionaremos o nome do arquivo `wp-config-sample.php` . Este será copiado para `wp-config.php` e ficará em `dest` .

Também vamos configurar `remote_src` com `yes` , assim evitaremos a busca na máquina de controle. Fizemos algo parecido com `unarchive` . Em seguida, consultaremos qual é a utilidade do `validade` na documentação. Trata-se de um comando usado para validar se um arquivo está funcionando, no caso, ele não nos será útil e será removido.

```
- copy:
    src: '/var/www/wordpress/wp-config-sample.php'
    dest: '/var/www/wordpress/wp-config.php'
    remote_src: yes
   become: yes
```

Observe que adicionamos `became` para conseguirmos fazer uma cópia do arquivo, lembrando que faremos isso como `root`. O próximo passo é rodarmos o playbook, no meu terminal, o retorno será `ok` porque já havia criado a cópia anteriormente.

Provavelmente, na sua máquina aparecerá `changed` pois você não criou a cópia do arquivo.

Nós vamos entrar na máquina virtual só para mostrarmos o arquivo e explicarmos como faremos essa configuração, entrando no diretório do Wordpress. Encontraremos `wp-config.php` na listagem:

```
vagrant@vagrant-ubuntu-trusty-64:~$ cd /var/www/wordpress
vagrant@vagrant-ubuntu-trusty-64:/var/www/wordpress$ ls
vagrant@vagrant-ubuntu-trusty-64:/var/www$ ls wordpress/
index.php     wp-active.php      wp-comments-post.php   wp-cron.php        wp-links.opml.php
license.txt   wp-admin           wp-config.php          wp-cron.php        wp-load.php
readme.html   wp-blog-header.php wp-config-sample.php  wp-includes          wp-login.php
```

Depois, rodaremos `less` no arquivo, e seremos informados que ele é um script PHP, no qual se definiu diversas constantes solicitadas pelo Wordpress.

```
vagrant@vagrant-ubuntu-trusty-64:/var/www/wordpress$ less wp-config.php

<?php
/**
 * The base configuration for Wordpress
 *
 * The wp-config.pho creation script uses this file during the
 * installation. You dont´t have to use the web site, you can
 * copy this file to "wp-config.php" and fill in the values.
 *
 * This File contains the following configurations:
 *
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package Wordpress
 */
```

As constantes de banco de dado já foram definidas e são essas as variáveis que iremos alterar.

```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_Name', 'database_name_here');

/** MySQL database username */
define('DB_USER', 'username_hostname here');

/** MySQL database password */
define(DB_PASSWORD', 'password_here');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database table. */
```

A seguir, usaremos uma estratégia de **expressões regulares** para alterarmos três valores:

- Nome do banco de dados: `database_name_here` ;
- Nome do usuário: `username_here` ;
- Senha: `password_here` .

Antes, descobriremos qual módulo nos apoiará a fazer as alterações dentro do arquivo. O módulo `replace` nos permitirá mudar a entrada de uma string por outra qualquer, dentro do arquivo. Ele vai alterar todas as ocorrências de uma determinada string por outra, é uma forma simples de fazermos expressões regulares.

Encontraremos um exemplo na documentação que atende as nossas necessidades:

```
- replace:
    path: /etc/hosts
    regexp: '(\s+)old\.host\.name(\s+.*)?$'
    replace: '\1new.host.name\2'
    backup: yes
```

Iremos adicioná-lo abaixo de `copy` da seguinte maneira:

```
- copy:
    src: '/var/www/wordpress/wp-config-sample.php'
    dest: '/var/www/wordpress/wp-config.php'
    remote_src: yes
  become: yes

- name: 'Configura o wp-config com as entradas do banco de dados'
  replace:
    path: '/var/www/wordpress/wp-config.php'
    regexp: '(\s+)old\.host\.name(\s+.*)?$'
    replace: '\1new.host.name\2'
    backup: yes
```

Em `regexp` adicionaremos a expressão regular. Se analisarmos o que é criado na máquina e ver como fica essa configuração de banco de dados dentro do `wp-config` , teremos o seguinte:

```
/** MySQL database password */
define(DB_PASSWORD', 'password_here');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');

/** The Database Collate type. Dont´t change this if in doubt.*/
define('DB_COLLATE', '');
```

Nós podemos usar essas expressões como os alvos de alteração da nossa expressão regular, podemos fazer isso de uma vez utilizando o parâmetro `with_items` abaixo de `backup` . Passaremos o objeto `regex` , composto por dois campo:

- `database_name_here`
- `value: 'wordpress_db` .

Considerando que o segundo é o nome que eu dei para o banco de dados. Em seguida, adicionaremos outras expressões `regex` .

```
with_items:
  - { regex: 'database_name_here', value: 'wordpress_db'}
  - { regex: 'username_here', value: ' wordpress_use'}
  - { regex: 'password_here', value: '12345'}
```

Após a adição de `with_items` , incluiremos `{{ item_regex}}` e `{{ item_value }}` e ficará da seguinte maneira.

```
- name: 'Configura o wp-config com as entradas do banco de dados'
  replace:
    path: '/var/www/wordpress/wp-config.php'
    regexp: "{{ item.regex }}"
    replace: "{{ item.value }}"
    backup: yes
  with_items:
    - { regex: 'database_name_here', value: 'wordpress_db'}
    - { regex: 'username_here', value: 'wordpress_user'}
    - { regex: 'password_here', value: '12345'}
  become: yes
```

Lembre-se que `item` é referente a uma entrada da lista que passamos no `with_items` . Como usamos um objeto com dois atributos (`regex` e `value`), quando o elemento `regex` for iterado, será alterado o valor dele, e o mesmo acontecerá com cada uma das entradas no campo `regex` . Por último, adicionamos `become: yes` e estamos prontos para testar.

No terminal, rodaremos o playbook.

```
$ ansible-playbook -i hosts provisioning.yml
```

As alterações serão bem-sucedidas.

```
TASK [ Descompacta o wordpress] ************************************************
ok: [172.17.177.40]

TASK [copy] ********************************************************************
changed: [172.17.177.40]


TASK [Configura o wp-config com as entradas do banco de dados] *****************
changed: [172.17.177.40] => (item={u'regex' : u'database_name_here', u'value': u'wordpress_db'})
changed: [172.17.177.40] => (item={u'regex' : u'username_here', u'value': u'wordpress_use'})
changed: [172.17.177.40] => (item={u'regex' : u'password_here', u'value': u'12345'})


PLAY RECAP *********************************************************************
172.17.177.40              : ok=8    changed=2    unreachable=0    failed=0
```

Depois, verificaremos como ficou o nosso arquivo.

```
$ vagrant ssh
```

Também usaremos o comando `less` :

```
vagrant@vagrant-ubuntu-trusty-64:~$ cd /var/www/wordpress/
vagrant@vagrant-ubuntu-trusty-64:/var/www/wordpress$ less wp-config.php

<?php
/**
 * The base configuration for Wordpress
 *
 * The wp-config.pho creation script uses this file during the
 * installation. You dont´t have to use the web site, you can
 * copy this file to "wp-config.php" and fill in the values.
 *
 * This File contains the following configurations:
 *
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package Wordpress
 */
```

Por último, teremos `define('DB_NAME', 'wordpress_db').

```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_Name', 'wordpress_db');
```

```
/** MySQL database username */
define('DB_USER', 'wordpress_use');

/** MySQL database password */
define('DB_PASSWORD', '12345');

/** MySQL hostname */
:
```

O Wordpress está pronto para se comunicar com o banco de dados? Responderemos isso a seguir, também vamos configurar o Apache, um passo que ainda está faltando.