04

Gerando configurações com templates

Transcrição

Nós configuramos diversas variáveis que nos ajudaram a melhorar a organização do código, que continha muita repetição. Nós também criamos o diretório `group_vars`, movemos para ele os arquivos que passam algumas variáveis úteis para o playbook e eliminamos os valores repetidos - incluindo, o diretório de instalação do Wordpress. Mas usamos este diretório também em `000-default.conf`, a configuração do Apache.

Como faremos para que o **Apache se aproveite do setup** das nossas variáveis, e não corrermos o risco de quebrarmos a instalação do Apache se mudarmos o diretório de instalação do Wordpress? Temos algumas variáveis dentro dos arquivos, desde que eles sejam templates. Estes são arquivos que servem de modelo para o Ansible aplicar modificações, ou para especificar algumas regras com a finalidade de gerar arquivos.

Os templates são manipulados por meio de um módulo homônimo. Por isso, nós vamos substituir a cópia de arquivo que fizemos para o playbook, por um template que vai gerar um arquivo de resultado direto na nossa máquina de destino. Como faremos isso? Em `provisioning.yml`, onde fazemos a cópia, incluiremos outro código de módulo, que terá como nome `Configura Apache para servir Wordpress`, com módulo `template`, originário da pasta `templates`. Vale ressaltar que todo template termina com a extensão `j2`.

```
- name: ' Configura Apache para servir Wordpress'
  template:
    src: 'templates/000-default.conf.j2'
    dest: '/etc/apache2/sites-available/000-default.conf'
  become: yes
  notify:
    - restart apache
```

O destino será o mesmo da task `Configura Apache para servir o Wordpress`, localizada logo abaixo. O próximo passo será salvar o arquivo na pasta `templates`, recém criada. Lembre-se de modificar a extensão para `j2`. Após a alteração, teremos o arquivo: `templates/ 000-default.conf.j2`. Nele, passaremos a variável `wp_installetion_dir`, exatamente como é feito no Ansible.

- Nós usamos o placeholder, assim ele fará a **substituição do valor**.

Quando precisarmos mudar o valor no arquivo `all.yml`, onde estão localizadas as variáveis, a alteração será feita automaticamente no arquivo recém criado.

Desta forma, evitamos erros e que o conteúdo do seu site fique quebrado. Em seguida, rodaremos o playbook e verificaremos se está tudo funcionando corretamente.

Nenhuma alteração será feita no banco de dados, mas o template será aplicado:

```
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'})
changed: [172.17.177.40] => (item={u'regex' : u'localhost', u'value': u'172.17.177.42'})
```

```
TASK [Configura Apache para servir o Wordpress ] ********************************************
ok: [172.17.177.40]

PLAY RECAP ***********************************************************************************
172.17.177.40              : ok=7    changed=2    unreachable=0    failed=0
172.17.177.42              : ok=5    changed=0    unreachable=0    failed=0
```

Se testarmos no navegador, veremos que a página de instalação do Wordpress será exibida. Tudo funcionou conforme o esperado.

Adicionamos o `template`, que será responsável por gerar o arquivo, está usando o valor da variável `wp_installation_dir` que está em `all.yml`, dentro da pasta `group_vars`.

Nós poderíamos ter feito também a parte do `replace`, no `provisioning.yml`. Teríamos copiado o arquivo de configuração do Wordpress e colocado todos os placeholders e usado `template`. Inclusive, esta poderia ter sido uma **solução melhor**, porque seria desnecessário colocar a declaração do que está sendo modificado dentro do código de `tasks`.

Assim, teríamos **separado as responsabilidades**: o template saberia o que deve ser substituído e as tasks saberiam o que deve ser executado e substituído.

Da forma como criamos a saída utilizando `replace`, as duas **informações ficaram misturadas**. Isso pode trazer problemas se você decidir refatorar o código, ou seja, prejudicamos a manutenibilidade conforme o código crescer. Porém, para fazermos substituições pouco complexas, trata-se de uma solução simples.

O `replace` pega o placeholder e substitui em **todas as ocorrências**.

Imagine a situação na qual substituiremos uma quantidade enorme de vezes uma string. Se fizéssemos isso com `template`, teríamos que colocar o placeholder na mesma quantidade de alterações necessárias, ainda que fossem `300` vezes.

Isso seria **diferente** com `replace`. Com este, colocaríamos apenas uma vez, como foi feito no nosso código. Desta forma já teríamos garantido a substituição.

A seguir, falaremos sobre roles.