02

Playbooks reaproveitáveis com Roles

Transcrição

Nós configuramos o Wordpress inteiro e o Ansible já está construindo toda nossa arquitetura de aplicação de três camadas - com Web server, banco de dados, Web Application de exemplo (no caso, Wordpress). Organizamos o código com o uso de variáveis, falamos sobre template para vermos como trazer essa organização de código com uso de variável para dentro também dos arquivos de configuração, que podemos gerar ou alterar.

A seguir, falaremos sobre como dividir melhor o nosso código e aumentar o seu reuso. O playbook faz o que prometeu, instala a aplicação de três camadas, porém, a capacidade de utilizarmos esse código novamente é baixa (talvez, zero).

Como o Ansible resolve isso? Ele nos permite criar pequenas bibliotecas de playbooks chamadas **roles**. Nós criaremos um grupo de três roles:

- Uma será responsável por instalar o Webserver;
- Uma para instalar o MySQL;
- E uma para instalar o Wordpress.

Nós teremos que reorganizar o código, quase não adicionaremos código. Começaremos pelo código do MySQL. Todas as roles têm um diretório chamado `tasks` e dentro, possuem um arquivo `main.yml` . Trata-se do arquivo padrão de execução das tasks da role, ou seja, é o ponto de entrada.

Iniciaremos o novo arquivo com os três traços habituais (`---`). E copiaremos os textos das tasks referentes a parte do hosts do MySQL para lá.

```
---

- name: 'Instala pacotes de dependencia do sistema operacional'
  apt:
    name: "{{ item }}"
    state: latest
 become: yes
 with_items:
  -mysql-server-5.6
  - python-mysqldb

- name: 'Cria o banco do MySQL'
  mysql_db:
    name: "{{ wp_db_name }}"
    login_user: root
    state: present

- name: 'Cria o usuário do MySQL'
  mysql_user:
    login_user: root
    name: "{{ wp_username }}"
    password: "{{ wp_user_password }}"
    priv: "{{ wp_db_name }}.*:ALL"
    state: present
```

```
      host: "{{ item }}"
    with_items:
     - 'localhost'
     - '127.0.0.1'
     - "{{ wp_host_ip }}"

- name: 'Configura MySQL para aceitar conexões remotas'
  copy:
    src: 'files/my.cnf'
    dest: '/etc/mysql/my.cnf'
  become: yes
  notify:
    - restart mysql
```

Temos as taks responsáveis por instalar os pacotes, criar o banco de dados, como configurar, incluímos o `notify` para o `handler`. Onde vamos adicionar o handler criado? As roles também têm um diretório específico para handlers, que receberá o nome de `handlers`:

roles/mysql/handlers

Em seguida, copiaremos o código do handler:

```
---
- name: restart mysql
  service:
    name: mysql
    state: restarted
  become: yes
```

Após separarmos o código referente ao banco de dados, removeremos o código copiado para outros arquivos e no lugar, invocaremos a role.

```
---
- hosts: database
  roles:
    - mysql

- hosts: wordpress
  handlers:
   - name: restart apache
     service:
       name: apache2
       state: restarted
     become: yes
```

Passamos a **lista de roles** que serão executados, testaremos para ver se o código continua funcionando. Quando executamos via role, a saída da task terá o nome da role, antes do texto adicionado no `name`, como vemos abaixo:

```
TASK [Gathering Facts] ********************************************************************
```

```
ok: [172.17.1777.42]

TASK [mysql : instala pacotes de dependencia do sistema operacional] ************************
ok: [172.17.177.42] => (item=[u'mysql-server-5.6'. u'python-mysqldb])

TASK [mysql : Cria o banco do MySQL] ********************************************************

TASK [mysql : Cria o usuário do MySQL] ******************************************************
ok: [172.17.1777.42] => (item=localhost)
ok: [172.17.1777.42] => (item=127.0.0.1)
ok: [172.17.1777.42] => (item=172.17.177.40)
```

Sabendo disso, temos como identificar.

A primeira role está pronta, em seguida, faremos a role do web server. Criaremos o arquivo `main.yml`, dentro de "roles > webserver > tasks". Depois, copiaremos o trecho de código referente à instalação do PHP e o Apache que está em `provisioning.yml`.

```
---
- name: 'Instala pacotes de dependencia do sistema operacional'
  apt:
    name: "{{ item }}"
    state: latest
  become: yes
  with_items:
    - php5
    - apache2
    - libapache2-mod-php5
    - php5-gd
    - libssh2-php
    - php5-mcrypt
    - php5-mysql
```

Este é o web server padrão. Como já movemos o trecho para o novo arquivo, podemos removê-lo do arquivo de origem e incluir `webserver` na lista roles:

```
---
- hosts: wordpress
  handlers:
   - name: restart apache
     service:
       name: apache2
       state: restarted
     become: yes
  roles:
    - webserver
```

A seguir, vamos executar o playbook. No retorno, veremos que será executada a instalação de pacotes.

```
TASK [webserver : Instala pacotes de dependencia do sistema operacional] *********************
ok: [172.17.177.42] => (item=[u'php5' , u'apache2', u'libapache2-mod-php5', u'php5-gd', u'libssl
```

```
TASK [Baixa o arquivo de instalacao de Wordpress] ***************************************
ok: [172.17.177.40]
```

Observe que nós misturamos a execução de roles com a de tasks, mas o Ansible sabe organizar isso. A execução acontecerá na ordem em que for passada.

Agora, criaremos a role final, que instalará o Wordpress. Em sua aplicação Web, você terá uma role para as dependências dela e terá outra, onde estará o código. Desta forma, quando for desenvolver uma nova aplicação usando o mesmo stack, você não precisará reescrever essa parte do código, apenas será necessário escrever o código da nova aplicação.

Começaremos criando o seguinte arquivo:

roles/wordpress/tasks/main.yml

Moveremos para esse arquivo o restante do código de `provisioning.yml` .

```
- hosts: wordpress
  handlers:
    - name: restart apache
      service:
        name: apache2
        state: restarted
      become: yes

  tasks:
    - name: 'Instala pacotes de dependencia do sistema operacional'
      apt:
        name: "{{ item }}"
        state: latest
      become: yes
      with_items:
        - php5
        - apache2
        - libapache2-mod-php5
        - php5-gd
        - libssh2-php
        - php5-mcrypt
        - php5-mysql

    - 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/
        remote_src: yes
      become: yes
```

```
    - 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: "{{ item.regex }}"
        replace: "{{ item.value }}"
      with_items:
        - { regex: 'database_name_here', value: 'wordpress_db'}
        - { regex: 'username_here', value: 'wordpress_user'}
        - { regex: 'password_here', value: '12345'}
      become: yes

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

Este trecho é responsável por baixar, descompactar, copiar arquivo de `config`. Também fizemos os replaces necessários para o funcionamento, fizemos o Apache reconhecer o Wordpress e reiniciamos o Apache. A seguir, vamos precisar do handler, que ficará em um arquivo que iremos gerar a seguir:

```
roles/wordpress/handlers/main.yml
```

Nele, adicionaremos `restart apache`:

```
---
- name: restart apache
  service:
   name: apache2
   state: restarted
  become: yes
```

Após salvarmos o handler e as tasks, nós podemos remover o trecho equivalente em `provisiong.yml`, que passará a ter somente:

```
---
- hosts: database
  roles:
   - mysql

- hosts: wordpress
  roles:
```

```
    - webserver
    - wordpress
```

Ao executarmos, veremos as duas roles serem listadas em sequência e que foram feitas as modificações necessárias.

Falta ainda copiarmos os arquivos de `template` de File, que usamos durante a criação do playbook para dentro das roles. Se optarmos por deixar o projeto como está, ele funcionará, porque o Ansible procura os arquivos de maneira hierárquica, então, ele procurará primeiramente dentro da role, depois, ele buscará "por fora", no nível do playbook. Esta não é a melhor prática.

Se decidirmos no futuro usar a role do MySQL em outro projeto, se não tivermos o arquivo do `my.cnf`, teremos problemas. Então, na role do `mysql`, vamos criar um diretório chamado `files` e moveremos `my.cnf` para dentro deste.

Em seguida, copiaremos o template `000-default-conf.j2` para a role do `wordpress`. Antes, criaremos um diretório chamado `templates` e nele será guardado o template. Observe que usamos a estrutura diretório `templates` para colocarmos os *templates files*, disponibilizados aos arquivos que só faremos cópia simples. E o Ansible consegue colocar esses arquivos dentro das roles.

O próximo passo será rodarmos o playbook, ele continuará fazendo as cópias, e os templates foram aplicados onde eram necessários. Agora, podemos apagar o diretório `templates` que ficou vazio.

Nossas roles estão completas, já especificamos o que elas devem cuidar. Elas guardam todos os arquivos necessários e o playbook se tornou apenas uma lista de encadeamento de roles, chamando-as quando são necessárias.

Mas falta adicionarmos um controle de comportamento mais específico, para as nossas roles. O que vai acontecer se chamamos a role do Wordpress, sem chamar de `webserver`, quando estamos rodando o Ansible pela primeira vez? Provavelmente, quebraremos a configuração da máquina e a role não chegará no estado desejado.

Ela vai depender do Apache e do PHP, ou seja, ela depende de coisas que a role de Webserver faz. O Ansible resolverá isso com um controle de dependência de roles.

E o que acontece se o usuário esquecer de passar alguma variável de configuração e que é usada dentro das roles? Elas não executarão com sucesso e o Ansible também nos dá uma maneira de determinarmos quais são os comportamentos padrões das roles, caso o usuário se esqueça de informar um valor. Veremos esses dois tópicos mais adiante.