06

Reiniciando serviços com Handlers

Transcrição

Nós vimos como configurar o Wordpress para encontrar o banco de dados dentro do MySQl e como se conectar a ele. Mostramos qual usuário deve ser usado, modificando `wp-config` e utilizando o módulo `replace`. O que faremos a seguir é preparar o Apache para encontrar o Wordpress e saber como servi-lo.

Começaremos com a localização do arquivo Apache, usado para servir o Wordpress. Entrando na nossa máquina virtual, acessaremos o caminho `etc/apache2/sites-available/000-default.conf` - este é o arquivo padrão usado pelo Apache para servir.

```
vagrant@vagrant-ubuntu-trusty-64:~$ cat /etc/apache2/sites-available/000-default.conf
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request´s Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not devisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicity.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
```

Acima vemos um trecho do conteúdo do arquivo que vai ser copiado, e criaremos uma cópia local que será modificada e ficará dentro da máquina virtual. Todos os arquivos com essa características ficam em um diretório chamado `files` e será nela, que colocaremos `000-default.conf`. Nós vamos jogar a cópia que fizemos do arquivo do Apache e modificaremos o caminho, adicionando o caminho no qual instalamos o wordpress:

```
DocumentRoot /var/www/wordpress
```

```
vagrant@vagrant-ubuntu-trusty-64:~$ cat /etc/apache2/sites-available/000-default.conf
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request´s Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not devisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicity.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/wordpress
```

O *Virtual Host* é a maneira como é dito para o Apache como ele deve servir para um determinado site. Quando especificamos `*:80`, indicamos que qualquer caminho passado para o apache na porta `80`, será realizada uma busca

no documento `/var/www/wordpress`, a melhor forma de servir. Este é o caminho da nossa instalação do Wordpress.

O próximo passo será copiá-lo para dentro da máquina virtual, usando novamente o módulo `copy` e a fonte (arquivo original) será `files/000-default.conf`. O destino será `/etc/apache2/sites-available/000-default.conf`.

Como estamos modificando um arquivo de configuração do sistema, então, ele deve ser feito como root. Faremos algo diferente ao uso anterior: como estamos copiando um arquivo que não está no servidor remoto, estamos tirando da nossa máquina de controle e empurrando para o servidor remoto, nós não usaremos a opção `remote_src` configurada com `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 }}"
    backup: yes
  with_items:
    - { regex: 'database_name_here', value: 'wordpress_db`}
    - { regex: 'username_here', value: 'wordpress_user'}
    - { regex: 'password_here', value: '12345'}
  became: yes

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

Para ver se funcionou, teremos que sair da máquina virtual e, depois, rodar o comando `playbook`. Ele fará novamente o passo a passo e o arquivo 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_user'})
changed: [172.17.177.40] => (item={u'regex' : u'password_here', u'value': u'12345'})

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

PLAY RECAP *************************************************************************************
172.17.177.40              : ok=9    changed=3    unreachable=0      failed=0
```

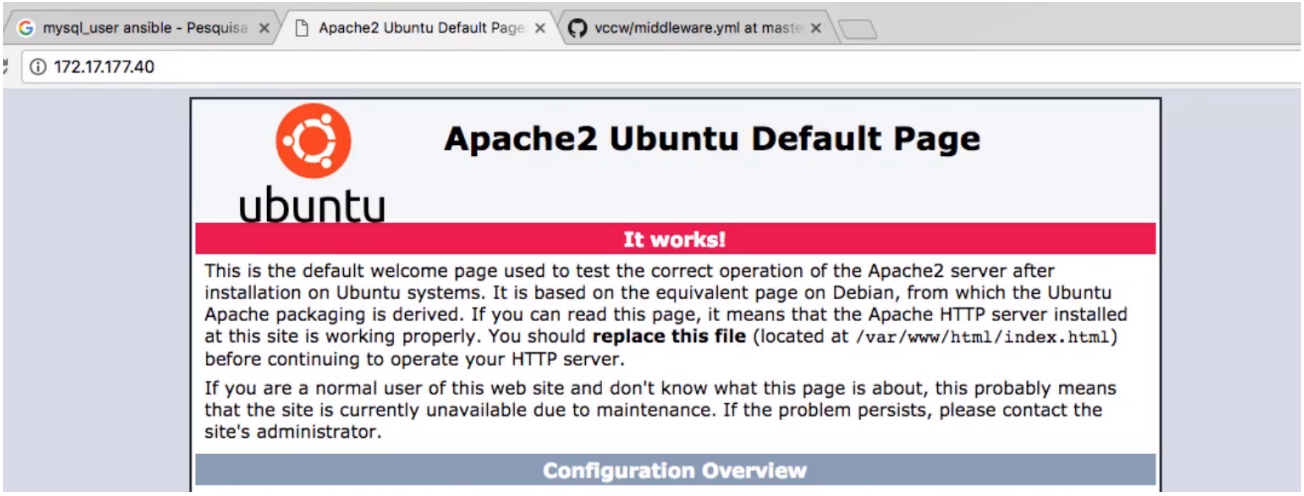
Com o comando `cat`, verificaremos se tudo funciona bem na máquina virtual.

```
vagrant@vagrant-ubuntu-trusty-64:~$ cat /etc/apache2/sites-available/000-default.conf
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request´s Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not devisive as it is used as a last resort host regardless.
```

```
        # However, you must set it for any further virtual host explicity.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/wordpress
```

Tivemos o retorno esperado, ele indica que `/var/www/wordpress` está onde deveria estar. Agora estamos prontos para servir o conteúdo do Wordpress, no entanto, se colocarmos o IP da máquina virtual na barra de navegação do browser, veremos a pagina padrão do Apache.



Por que ele não está servindo o Wordpress? Isto aconteceu porque alteramos a configuração do Apache, porém, ele não foi reiniciado. O Ansible deve ser capaz de avisar ao Apache que o conteúdo foi alterado, desde que seja relevante e assim, deva ser reiniciado.

Uma solução é usando ***handlers***, que são tasks Ansible simples, porém, elas podem ser chamadas com notificações e em momentos diversos do seu código. É desnecessário que ela esteja em uma linha de execução pré-definida como é feito do script do Playbook. Veremos a seguir como elas são feitas.

De volta ao arquivo `provisoning.yml` , no início, criaremos um campo cujo nome será `handlers` e escreveremos a task no mesmo nível de `tasks` .

```
- hosts: all
  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
        - mysql-server-5.6
        - python-mysqldb
        - php5-mysql
```

A nova task recebeu o nome de `restart apache` e, o módulo se chama `service` - utilizado para administrar sistemas Demo do Linux, como *systemd*, *upstart*, *init.d*. Os parâmetros que passaremos para o service será `apache2` , você deve saber o nome do serviço que será manipulado. O state configurado `restarted` significa que se chamarmos esse handler, ele deve reiniciar o Apache. E como trabalhamos com um serviço adminstrativo, ele será feito como root.

O próximo passo será alterarmos a task que modifica o arquivo de configuração do Apache e dentro de `notify` , chamaremos uma lista de *handlers*. Estes deverão ser disparados, no caso, queremos trabalhar com `restart apache` .

```
- 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
```

Em seguida, rodaremos o playbook. Após executarmos o comando, veremos que não conseguiremos rodar o playbook porque faltou modificar o arquivo.

```
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'})

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

PLAY RECAP ************************************************************************************
172.17.177.40              : ok=9    changed=3    unreachable=0      failed=0
```

Mas vemos que o Apache foi configurado. Para resolvermos o assunto, faremos uma pequena "trapaça". Dentro de `provisioning.yml` , adicionaremos o texto `iii` no comentário, assim justificaremos modificação do arquivo e ser recarregado no servidor.

```
# Available loglevels: iii trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It os also possible to configure to configure the loglevel for particular
# modules, e.g.
# LogLevel info ssl>warn
```

Desta vez, a cópia será realizada e o handler, disparado.

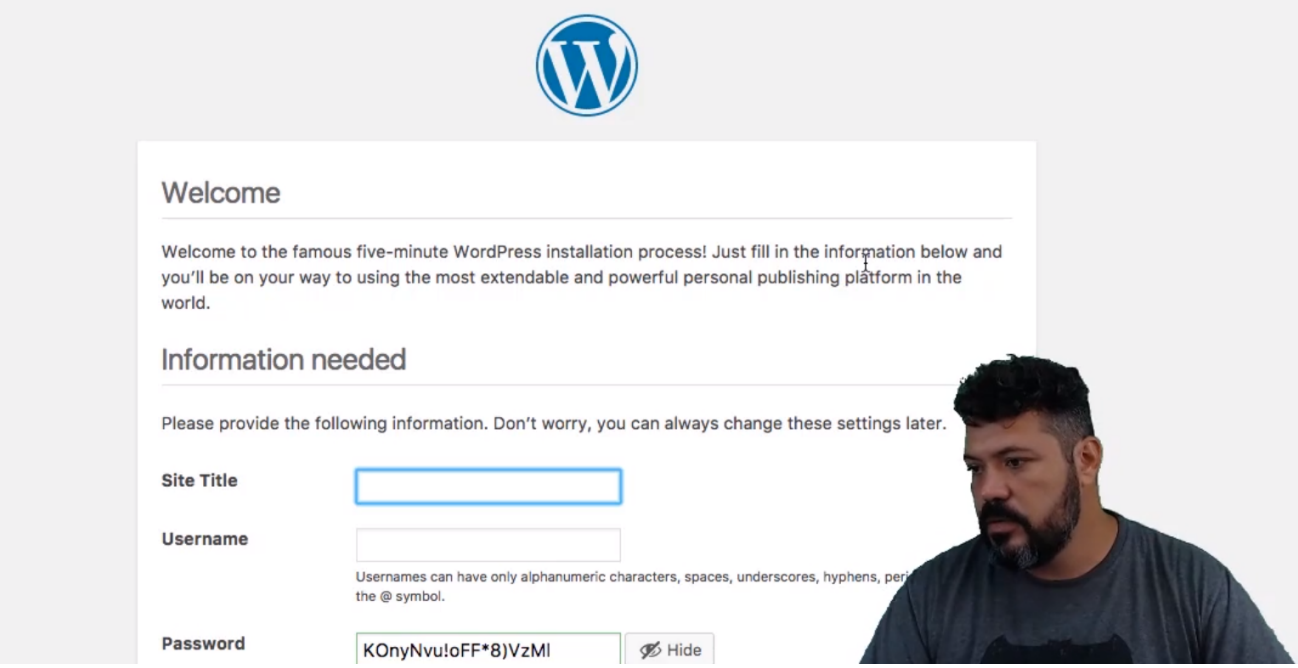
```
TASK [Configura o wp-config com as entradas do banco de dados] ********************************
changed: [172.17.177.40] => (item={u'regex' : u'database_name_here', u'value': u'wordpress_db'})
changed: [172.17.177.40] => (item={u'regex' : u'username_here', u'value': u'wordpress_user'})
changed: [172.17.177.40]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

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

RUNNING HANDLER [restart apache] ***************************************************************
changed: [172.17.177.40]

PLAY RECAP *************************************************************************************
172.17.177.40              : ok=10    changed=4    unreachable=0      failed=0
```

Agora, esperamos que tudo esteja funcionando corretamente.



Teremos que configurar a instalação de Wordpress, com o nome do seu site e usuário. Todos os passos seguidos até este momento, serão os mesmos passos que você vai utilizar para instalar qualquer aplicação que use o modelo de três camadas, na qual você tem o banco de dados, um servidor web e a aplicação em si. Caso esteja trabalhando com outro objeto open source, seja phpBB, Magento, outra tecnologia que não trabalhe com PHP, também funcionará.

A diferença será na seleção de quais pacotes serão instalados, seja da plataforma ou web server. Às vezes, trabalhamos com Nginx, com Tomcat, mesmo assim, teremos que instalar os pacotes da maneira como a distribuição escolhida funcionará. Instalaremos o banco de dados, seja MySQL, PostgreSQL, ou a tecnologia mais apropriada.

Lembre-se que ainda será preciso fazer a configuração, criar um usuário para a aplicação, dar as permissões corretas - o usuário tem que ter ações limitadas. A aplicação, baixada da internet - como foi o nosso caso -, fica em um repositório privado que você usa no seu trabalho, em casa ou de um servidor de integração.

Você também terá que baixá-la e descompactá-la, deverá saber como fazer a instalação. Teremos que configurar qualquer aplicação - a menos que você opte deixar *hardcode*, uma prática pouco recomendada.

Você deverá fazer o servidor web saber onde está sua aplicação, devemos fazer os dois se comunicarem. Logo, os passos visto no curso, abrangem o *baseline* necessário para fazer uma aplicação de três camadas funcionar, independente da tecnologia usada. Se ele rodar com Linux, o Ansible serve como base de configuração qualquer tecnologia.

A seguir, aprenderemos a usar este conteúdo em produção, porque o que fizemos ainda parece com ambiente de desenvolvimento. Instalamos tudo na mesma máquina, mas por que precisaríamos separar a aplicação web do banco de dados? Por questões de segurança, de performance, preciso porque aplicações webs são diferentes de bancos de dados.

Provavelmente, serão necessários mais nós de um aplicação web, do que de um banco de dados em produção. Você precisa ter uma forma de separar as responsabilidades dos seus servidores. Faremos isso mais adiante.