06

Mãos na massa: Fazendo a separação do banco e da aplicação

Chegou a hora de você executar o que foi visto na aula! Para isso, execute os passos listados abaixo.

1) Para separar o banco de dados da aplicação, primeiramente crie duas máquinas virtuais, alterando o **Vagrantfile**:

```
Vagrant.configure("2") do |config|

  config.vm.box = "ubuntu/trusty64"

  config.vm.provider "virtualbox" do |v|
    v.memory = 1024
  end

  config.vm.define "wordpress" do |m|
    m.vm.network "private_network", ip: "172.17.177.40"
  end

  config.vm.define "mysql" do |m|
    m.vm.network "private_network", ip: "172.17.177.42"
  end

end
```

2) Em **hosts**, adicione a máquina nova, separando em um novo grupo, o grupo **database**:

```
[wordpress]
172.17.177.40 ansible_user=vagrant ansible_ssh_private_key_file="/Users/marcoscropalato/wordpres

[database]
172.17.177.42 ansible_user=vagrant ansible_ssh_private_key_file="/Users/marcoscropalato/wordpres
```

3) Feito isso, crie a máquina virtual nova, executando o seguinte comando no terminal:

```
vagrant up mysql
```

4) Em **provisioning.yml**, separe os comandos, pois agora você terá um servidor especializado no banco de dados e um servidor especializado na aplicação. Então, crie um grupo de *hosts* novo, com o mesmo nome do grupo colocado no arquivo **hosts**. Nesse grupo, você instalará os pacotes **mysql-server-5.6** e **python-mysqldb**, e configurará o banco de dados, criando o *database* e o usuário do MySQL:

```
- hosts: database
  tasks:

    - 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: wordpress_db
      login_user: root
      state: present

  - name: 'Cria o usuário do MySQL'
    mysql_user:
      login_user: root
      name: wordpress_user
      password: 12345
      priv: 'wordpress_db.*:ALL'
      state: present
```

5) Os comandos de instalação do PHP, WordPress e Apache não devem ser executados nessa máquina. Então, modifique o **hosts** `all` para somente `wordpress` .

6) Agora, rode o *Playbook*, executando o seguinte comando no terminal:

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

7) Para testar se o banco de dados foi instalado na máquina nova, logue-se com o usuário criado no banco de dados:

```
vagrant ssh mysql
mysql -u wordpress_user -p12345
```

8) Veja os *databases* através do comando `show databases` e veja a base de dados criada

Criando a máquina do WordPress sem MySQL

9) Ainda em **provisioning.yml**, remova todos os comandos relacionados à instalação de banco de dados da máquina **wordpress**:

```
- 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
```

10) Feito isso, destrua essa máquina, crie uma nova e configure-a, executando no terminal:

```
vagrant destroy -f wordpress
vagrant up wordpress
ansible-playbook -i hosts provisioning.yml
```

11) No arquivo de configuração do WordPress, é preciso informar o *host* do banco de dados, alterando o valor `localhost` para o IP da máquina onde está rodando o banco de dados:

```
- 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'}
    - { regex: 'localhost', value: '172.17.177.42'}
  become: yes
```

12) Por padrão, a instalação do MySQL não aceita conexões que de outros IPs que não sejam *localhost* e os usuários criados só possuem permissão local, então essas duas informações precisam ser modificadas. Então, no *host* ***database***, quando o usuário do MySQL é criado, adicione uma lista de *hosts* possíveis de se conectar com o banco de dados:

```
- name: 'Cria o usuário do MySQL'
  mysql_user:
    login_user: root
    name: wordpress_user
    password: 12345
    priv: 'wordpress_db.*:ALL'
    state: present
    host: "{{ item }}"
  with_items:
    - 'localhost'
    - '127.0.0.1'
    - '172.17.177.40'
```

13) Para que o serviço do MySQL aceite conexões remotas, altere o seu arquivo de configuração. Execute os comandos abaixo para realizar a conexão com a máquina e visualizar o conteúdo do arquivo **my.cnf**. Ao visualizá-lo, copie o seu conteúdo:

```
vagrant ssh mysql
cat /etc/mysql/my.cnf
```

Copiado o conteúdo do arquivo **my.cnf**, dentro do seu projeto, na pasta **files**, crie um novo arquivo também com o nome **my.cnf**, e cole o conteúdo copiado anteriormente. Colado o conteúdo, altere o valor da propriedade **bind-address** para **0.0.0.0**:

```
bind-address        = 0.0.0.0
```

14) Agora, é preciso informar o Ansible para fazer a cópia deste arquivo:

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

```
    notify:
      - restart mysql
```

15) Para isso funcionar, é necessário reiniciar o serviço do MySQL, então crie um *handler* para tal e reinicie o serviço. Ao final, o **host** ficará assim:

```
- hosts: database
  handlers:
    - name: restart mysql
      service:
        name: mysql
        state: restarted
      become: yes

  tasks:
    - 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: wordpress_db
        login_user: root
        state: present

    - name: 'Cria o usuário do MySQL'
      mysql_user:
        login_user: root
        name: wordpress_user
        password: 12345
        priv: 'wordpress_db.*:ALL'
        state: present
        hosts: "{{ item }}"
      with_items:
        - 'localhost'
        - '127.0.0.1'
        - '172.17.177.40'

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

16) Apesar dos ajustes, ele não vai conseguir copiar novamente o arquivo my.cnf, porque ele já existe. Para que ele possa chamar o handler, faremos uma pequena alteração no arquivo, incluindo * após Basic Settings:

```
[mysqld]
#
# * Basic Settings *
#
```

17) Por fim, rode o *Playbook*:

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