4525 – Curso de Infraestrutura Ágil com Práticas DevOps

Desafio - Automação com Ansible

Neste desafio iremos utilizar os conhecimentos que tivemos na aula **Automação com Ansible e Gerenciando** este desafio será divido em duas parte, na primeira parte vamos desenvolver uma role do ansible para criação de virtualhost para o servidor Apache. Na segunda parte iremos desenvolver um playbook para provisionamento automático dos containers do Docker.

Parte 1. Trabalhando com roles e templates com Ansible

Na primeira parte do nosso desafio vamos configurar uma role do ansible para criação de virtualhost no apache, ao criar uma role do ansible, ela pode ser utilizada em vários projetos, habilitando o uso desta role para qualquer aplicação, ou serviço de automação que necessita de virtualhost a ser configurado e para atender ambientes a diferentes, como centos e debian iremos utilizar variáveis que podem ser alterados de acordo com a necessidade do ambiente.

1. Crie um container do Docker para realizar a homologação desta role. No próprio repositório do Docker, existe containers prontos com ansible instalado. Crie um container com o nome e hostname **ansible-hom** utilizando a imagem **ansible/ubuntu14.04-ansible.**

$ docker run -it --name ansible --hostname ansible ansible/ubuntu14.04-ansible /bin/bash

2. A roles é uma forma simples de dividir um playbook em diretórios e arquivos para melhorar o nível de gerenciamento das configurações do ansible e poder reutilizar o código em outros projetos de automação. Dentro do diretório /home crie um diretório

chamado ansible, e dentro do diretório realizar a criação da estrutura da nossa role descrita a baixo:

- roles/
  - apache/
    - templates/
    - tasks/
    - vars/
    - handlers/

templates: O diretório onde será armazenado o nosso arquivos de templates.

tasks: O diretório onde será armazenado as terefas da role.

vars: O diretório onde será armazenado as configurações das variaveis.

handlers: O diretório onde será armazenados o gerenciados de eventos como reiniciar um serviço depois de executar uma tarefa (tasks).

```
$ cd /home
$ mkdir apache/
$ cd apache
$ mkdir files handlers tasks templates vars
```

3. Os arquivos com o nome **main.yml** dentro dos diretórios da role são carregados automaticamente pelo playbook do ansible, isso significa que as taks da role serão colocadas dentro do arquivos **tasks/main.yml**, as variáveis da role será inseridas dentro do arquivo **vars/main.yml**, o gerenciadores de eventos são definidos dentro do diretório **handlers/main.yml**

3.1. Quando você utiliza variáveis em roles, você precisa passar as variáveis ao executar a role dentro de um playbook, e você pode definir alguns valores padrões para essas variáveis. Dentro do arquivo **vars/main.yml** realize a configuração das quatro variáveis a baixo:

- Uma variável para o domínio do virtualhost.
- Uma variável com o diretório de configuração do apache.
- Uma variável com o diretório dos arquivos de log do apache.
- Uma variável com o diretório dos virtualhosts do apache

OBS: Para distribuições Debian e Ubuntu segue a estrutura de diretório do apache2.

/var/log/apache2 → Diretório dos arquivos de log do apache.

/etc/apache2 → Diretório dos arquivos de configuração do apache.

/etc/apache2/sites-availables → Diretório dos arquivos de virtualhost do apache.

Ao configurar uma role com variáveis você disponibiliza a possibilidade de trabalhar com ambientes diferentes, nesse caso como exemplo: a diferença do do Debian e Centos são os diretórios onde ficam armazenados os arquivos de configuração do apache, ao utilizar a role basta mudar os variáveis para adaptar as configurações para cada ambiente, tornando a nossa role mais robusta e reutilizável em outros projetos.

Ao chamar o Playbook será obrigatório passar uma variável com nome do virtualhost, que será chamado a partir do plugin. O nome do virtualhost define o url do site, a pasta na qual o arquivos web estão armazenados e nome do arquivos de log. para o site webmail.dexter.com.br *o nome passado seria webmail de* exemplo.

```
$ vim vars/main.yml
---
# Vars: variaveis
```

domain: 'dexter.com.br'
logdir: '/var/log/apache2'
confdir: '/etc/apache2/'
vhostdir: "/etc/apache2/sites-available"

3.2. O Virtualhost para o apache é simplesmente um arquivo de texto dentro do diretório de configuração do Apache. Esse arquivo precisa ser dinâmico ou seja as informações que contém dentro dele precisa mudar de acordo com o nome do virtualhost, para criar arquivos dinâmicos utilizamos os templates.

Veja um exemplo do arquivo de template:

```
diretorio_conf = {{ confdir }}
arquivo_log = {{ logfile }}
```

Assim que ansible ler o arquivo do template através do modulo **template**, o ele ira trocar as variáveis do arquivos tornando assim o arquivo dinâmico. Crie o arquivo de template do virtualhost do apache dentro da pasta templates.

Segue o arquivo configurado de forma estática, a sua tarefa e trocar as informações por variáveis que definimos no exercícios anterior:

```
<VirtualHost *:80>
      ServerName webmail.dexter.com.br
      DocumentRoot /var/www/webmail
      ErrorLog /var/log/apache2/webmail-error.log
      CustomLog /var/log/apache2/webmail-access.log common
</VirtualHost>
```

$ vim templates/vhost.template
<VirtualHost *:80>
ServerName {{ name }}.{{ domain }}

```
  ServerAlias www.{{ name }}.{{ domain }}
  ErrorLog {{ logdir }}/{{ name }}-error.log
  CustomLog {{ logdir }}.{{ name }}-access.log common
</VirtualHost>
```

3.3. Assim que você alterar os arquivos de configuração é preciso reiniciar os serviços para aplicar as novas configurações para isso você pode utilizar a opção **notify** dentro das tasks para chamar um handler, criando um evento durante o processo de automação.

Exemplo de um handler:

handlers/main.yml:

```
- name: restart dns
  service: state=bind9 state=restarted
```

Usando o **notify** com o nome **restart dns** dentro das tasks do ansible, ao realizar a tarefa o ansible ira chamar o handler no caso do modulo service realizando um restart no serviço do bind.

Segue o exemplo:

tasks/main.yml

```
- name: copying files
  copy: src=dns-file.conf dest=/etc/bind/named.conf.options
  notify: restart dns
```

Crie o handler responsável por reiniciar o serviço do Apache dentro do arquivo handlers/main.yml.

```
$ vim handlers/main.yml
---
# Handlers: Identify the containers
```

```
- name: restart apache
  service: state=apache2 state=restarted
```

3.4. Para finalizar o role de criação do virtualhost do Apache, crie o arquivo **tasks/main.yml** com as tarefas de automação;

- Colocar o arquivo de template do virtualhost no diretório de configuração do virtualhost do apache:

- Habilitar o virtualhost do apache através do comando a2ensite <nome_do_vhost>.

OBS: Utilize os módulos command e template para realizar as tarefas.

```
$ vim tasks/main.yml
---
- name: Ensuring templates
  template: src=vhost.template dest={{ vhostdir }}/{{ name }}.conf

- name: Enable apache virtualhost
  command: a2enmod {{ name }}
  notify:
    - restart apache
```

Para chamar roles basta criar um playbook e na opção **roles** passar nome roles e seu parâmetros de configuração. Após criar a role dentro do diretório /home/ansible crie o arquivo webmail.yml com contéudo a baixo:

```
---
# playbook
- name: Garantir apache
  hosts: localhost
  connection: local

  roles:
    - { role: apache, name: webmail }
```

Segue a estrutura de diretório final desse desafio:

```
/home
   - ansible/
      - webmail.yml
      - roles/
           - apache/
                - templates/
                - handlers/
                - vars/
                - tasks/
```

Agora a primeira parte do desafio está criada, lembrando que realizar a configuração do arquivos do serviços, utilizar eventos para reiniciar o serviço é uma das tarefas mais usadas dentro de uma infraestrutura.

Parte 2. Provisionamento automático de containers do Docker

Com o uso docker ou outras ferramentas de virtualização e Computação em Nuvem, pode automatizar o processo de criação de máquina para o nosso ambiente. Na segunda parte do desafio vamos provisionar automaticamente o container no docker através do Ansible.

OBS: Realize a tarefa na máquina DevOps do curso

Segue a baixo as tarefas que o playbook responsável pela criação do container de homolog do Dns terá que realizar de forma automática.

1. Remover o container caso ele existe, você pode utilizar o comando **docker inspect** para verificar se container existe ou não. Lembrando que na aula nos vimos o parâmetro when que cria condições e depêndencias para as tarefas.

2. Crie o container de homologação do DNS novamente, utilizando as informações a baixo:

- **nome:** dns-homolog
- **hostname:** dns-homolog
- **image:** homolog-template
- **comando:** /bin/bash
- **Adcione** uma entrada no host com o nome puppet apontando para o endereço 192.168.200.50, utilizando o parâmetro –add-host na criação do container.

```
---
# Playbook:
- name: Run docker - Installed puppet agent and test service
  hosts: docker

  tasks:
  - name: Remove the container if already exists
    command: docker inspect ns-homolog
    register: container
    ignore_errors: yes

  - name: Stop the container before remove
    command: docker stop ns-homolog
    when: container.rc == 0

  - name: Remove the container where already exists
    command: docker rm ns-homolog
    when: container.rc == 0

  - name: Create the dns-teste container
    command: docker run -dit --name dns-homolog --hostname dns-homolog --add-host=puppet:192.168.200.50 homolog-template /bin/bash
```

Ao finalizar a tarefa o seu desafio estará completo. Esse tarefa será a base para utilizamos no laboratório final onde teremos o provisionamento das máquinas de homologação de forma automática, realizando o teste das configurações para validar as novas de configurações de forma automatizada, para evitar problema com as configurações no ambiente de produção.