4525 – Curso de Infraestrutura Ágil com Práticas DevOps

Desafio - Laboratório Parte 4: Automação da Aplicação Dexter

Neste desafio iremos utilizar os conhecimentos que tivemos no desenvolvimento do laboratório da Dexter para melhorar a configuração do containers. Durante todos laboratórios nos criamos uma playbook chamada **docker.yml** para cada serviço, sendo que esse arquivos sempre segue o mesmo padrão apenas mudando o nome do containers, que é especifico para cada serviço. Porém com arquivos configurados dessa forma, caso você precisa alterar alguma informações terá que alterar em cada, arquivo, e além do mais se você tiver muito mais servidores do que cinco que temos na Dexter.

Nesse desafio vamos configurar uma role que é responsável por provisionar o container, sendo nessa role nos podemos passar qual tipo de container ele vai, criar quais os nomes, configuração de IP entre outras informações, deixando a manutenção uma melhor manutenção caso precise, porque quando alterar a role servirá para todos os ambientes e deixará a configuração do ansible seguindo as boas práticas do Ansible.

Nessa roles deve aceitar os seguintes parâmetros e construir os containers de acordo com a variável:

- **docker_name:** Nome do container e do hostname da máquina
- **docker_ip:** Endereço ip fixo
- **docker_environment:** Produção ou Homologação
- **docker_image:** imagem que deve ser utilizado para a criação do container.

Quando a role do ansible for configurada com sucesso, agora temos a lógica de criação do containers do Docker, basta remover o arquivo docker.yml, e no main.yml realizar o include da nossa role para o us da mesma máquina.

Estrutura de diretório do roles

/etc/ansible/roles

→ roles/docker

→ tasks/

→ handlers/

→ vars/

A estrutura de diretório não existe, então é necessário realizar a criação deste diretório e configurar a role utilizando a mesma logica do nossos arquivos de Playbook dentro da Role.

DICA: Podemos utilizar a opção de condicional trabalhando em conjunto da opção de include do ansible, assim você realizar include quando alguma condição for passada ao playbook or roles, como por exemplo realizar o include de um arquivo caso o ambiente for de homologação, ou chamar outro arquivo caso o ambiente for de produção.

Precisamos definir a estrutura de pasta para a nossa role de nome docker

```
# cd /etc/ansible/roles/
# mkdir docker/{templates,tasks,files,handlers} -p
```

Vamos agora realizar a construção do arquivo main.yml dentro do diretório tasks da role, nesse arquivo será realizado o include para o arquivos responsável pela criação do ambiente de acordo com a variável passada. Seguindo a logica caso o ambiente for homolog, realizar o include do arquivo homolog.yml, caso o ambiente for production realiza o include do arquivo production.yml, sendo assim cada arquivo e responsável por

cada ambiente:

```
# vim docker/tasks/main.yml

---
# Importando o arquivo production.yml quando a variavel
# enviroment for definida como production
- include: production.yml
  when: docker_environment == "production"

# Importando o arquivo homolog.yml quando a variavel
# environment for definida como homolog
- include: homolog.yml
  when: docker_environment == "homolog"
```

A logica para criação para containers de produção continua a mesma vista durante o laboratório Dexter, porém agora o nome do container, a imagem utilizada e endereço são passados como variáveis, assim ao chamar a role você precisa sempre definir as variáveis que estão escritas nas arquivos de tasks (tarefas) do playbook:

```
# vim docker/tasks/production.yml

---
# Playbook: create docker container
- name: starting docker container
  command: docker start {{ docker_name }}
  ignore_errors: yes
  register: result

- name: creating docker container
  command: docker run -dit --name {{ docker_name }} --hostname {{ docker_name }} --add-host=puppet:10.8.0.6 --dns-search=dexter.com.br --net dexterlan --ip {{ docker_ip }}
```

```
--restart=always {{ docker_image }} /bin/bash
  when: result.rc != 0

- name: running puppet agent
  command: docker exec {{ docker_name }} puppet agent -t --environment=production
  ignore_errors: yes
```

A logica para criação para containers de homologação continua a mesma vista durante o laboratório Dexter, porém agora o nome do container, a imagem utilizada são passados como variáveis, assim ao chamar a role você precisa sempre definir as variáveis que estão escritas nas arquivos de tasks (tarefas) do playbook:

```
# vim docker/tasks/homolog.yml

---
# craete and test containers

- name: testing docker container
  command: docker inspect {{ docker_name }}
  register: container
  ignore_errors: yes

- name: stop docker container
  command: docker stop {{ docker_name }}
  when: container.rc == 0

- name: remove docker container
  command: docker rm {{ docker_name }}
  when: container.rc == 0

- name: create docker container
  command: docker run -dit --name {{ docker_name }} --hostname {{ docker_name }} --add-
```

```
host=puppet:192.168.200.50 {{ docker_image }} /bin/bash

- name: running puppet agent
  command: docker exec {{ docker_name }} puppet agent -t --environment=homolog
  ignore_errors: yes
```