Support pédagogique Ansible

Docker

Ansible et Docker

Installation de Docker

- On prépare le fichier inventaire :

```
[dockerhost]
1-2.practice-k8s.cloud
```

- On va installer un rôle pour installer Docker

```
$ mkdir docker && cd docker
```

```
$ ansible-galaxy  install geerlingguy.docker -p roles --force
```

```
Starting galaxy role install process
- downloading role 'docker', owned by geerlingguy
- downloading role from https://github.com/geerlingguy/ansible-role-docker/archive/6.0.3.tar.gz
- extracting geerlingguy.docker to /home/alex/ansible/2022/support/docker/roles/geerlingguy.docker
- geerlingguy.docker (6.0.3) was installed successfully
```

Installation de Docker

- Un autre rôle pour installer les `Python library for the Docker Engine API`

```
$ ansible-galaxy  install geerlingguy.pip -p roles/ --force
```

```
Starting galaxy role install process
- downloading role 'pip', owned by geerlingguy
- downloading role from https://github.com/geerlingguy/ansible-role-pip/archive/2.2.0.tar.gz
- extracting geerlingguy.pip to /home/alex/ansible/2022/support/docker/roles/geerlingguy.pip
- geerlingguy.pip (2.2.0) was installed successfully
```

> Jeff Geerling, author of Ansible for DevOps.

Installation de Docker

- On vérifie que les deux rôles ont bien été installés

```
$ ansible-galaxy role list --roles-path ./roles
```

```
# /~/docker/roles
- geerlingguy.docker, 7.4.7
- geerlingguy.pip, 3.1.0
```

Installation de Docker

- On créé le playbook pour installer `Docker` et les `liberairies Pyhon` pour Docker sur la cible

```
$ cat deploy-docker.yaml
```

```
---
- name: Deploy Docker to Host
  hosts: "{{ NODES }}"
  become: yes
  vars:
    pip_install_packages:
      - name: docker
  tasks:
    - name: Install docker
      include_role:
        name: geerlingguy.docker
    - name: Install Packges
      include_role:
        name: geerlingguy.pip
```

Installation de Docker

- Export le fichier de configurion Ansible

```
$ export ANSIBLE_CONFIG=/root/ansible-practice/ansible.cfg
```

- On lance le playbook

```
$ ansible-playbook deploy-docker.yaml -e "NODES=dockerhost"
```

```
1-2.practice-k8s.cloud     : ok=14   changed=7    unreachable=0    failed=0    skipped=14
```

- On vérifie que la cible que Docker est bien installé

```
$ ansible dockerhost -a 'docker version'
...
Version:          27.0.3
...
```

Gérer les containers avec Ansible

> Nous allons utiliser la collection `community.docker` (https://galaxy.ansible.com/community/docker), contient plus de 25 Ansible modules et ~10 plugins pour la connexion, inventory, et plus encore.

- On vérifie que la collection est dans la place

```
$ ansible-galaxy collection list | grep docker

community.docker 3.2.1
```

Gérer les containers avec Ansible

- On va créer un playbook pour gérer un container Nginx

```
$ cat container-manage.yaml
```

```
---
- name: Manage Docker containers
  hosts: "{{ NODES }}"
  become: yes
  vars:
    container_image: nginx
    container_name: web
    container_port: 80
    container_expose_port: 8080
    container_action: 'run'
```

```
tasks:
  - name: Create and Start a Docker container
    community.docker.docker_container:
      name: "{{ container_name }}"
      image: "{{ container_image }}"
      state: started
      ports: "{{ container_expose_port }}:{{ container_port }}"
    when: container_action == 'run'

  - name: Stop Docker container
    community.docker.docker_container:
      name: "{{ container_name }}"
      state: stopped
    when: container_action == 'stop'

  - name: Remove Docker container
    community.docker.docker_container:
      name: "{{ container_name }}"
      state: absent
    when: container_action == 'stop'
```

```
- name: Verify web site running inside container
  hosts: "{{ NODES }}"
  become: no
  vars:
    container_expose_port: 8080
    container_action: 'run'
  tasks:
    - name: Verify application health
      ansible.builtin.uri:
        url: http://{{ inventory_hostname }}:{{ container_expose_port }}
        status_code: 200
      delegate_to: localhost
      when: container_action == 'run'
```

Gérer les containers avec Ansible

```
$ ansible-playbook container-manage.yaml -e "NODES=dockerhost"
```

```
fatal: [1-2.practice-k8s.cloud]: FAILED! => {"changed": false, "msg": "Error
connecting: Error while fetching server API version: Not supported URL scheme http
+docker"}
```

> Vérification de la version du module Docker installé

```
$ ansible-galaxy collection list community.docker

  Collection       Version
  ---------------- -------
  community.docker 3.4.11
```

> Il faut upgrade la version de la `community.docker` (> 3.10)

```
$ ansible-galaxy collection install community.docker  --force
```

Gérer les containers avec Ansible

```
$ ansible-playbook container-manage.yaml -e "NODES=dockerhost"
```

```
TASK [Gathering Facts] *********************************************************
ok: [rocky-docker]

TASK [Create and Start a Docker container] *************************************
ok: [rocky-docker]

TASK [Stop Docker container] ***************************************************
skipping: [rocky-docker]

TASK [Remove Docker container] *************************************************
skipping: [rocky-docker]

PLAY [Verify web site running inside container] ********************************

TASK [Gathering Facts] *********************************************************
ok: [rocky-docker]

TASK [Verify application health] ***********************************************
ok: [rocky-docker -> localhost]

PLAY RECAP *********************************************************************
rocky-docker               : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0
```

Gérer les containers avec Ansible

- Le résultat donne :

```
$ curl http://1-2.practice-k8s.cloud:8080 | grep nginx
```

```
<title>Welcome to nginx!</title>
```

Stopping Docker

- Pour gérer les status d'un container, on va utiliser la variable `container_action` défini dans le `playbook`

```
$ ansible-playbook container-manage.yaml -e "NODES=dockerhost container_action=stop"
```

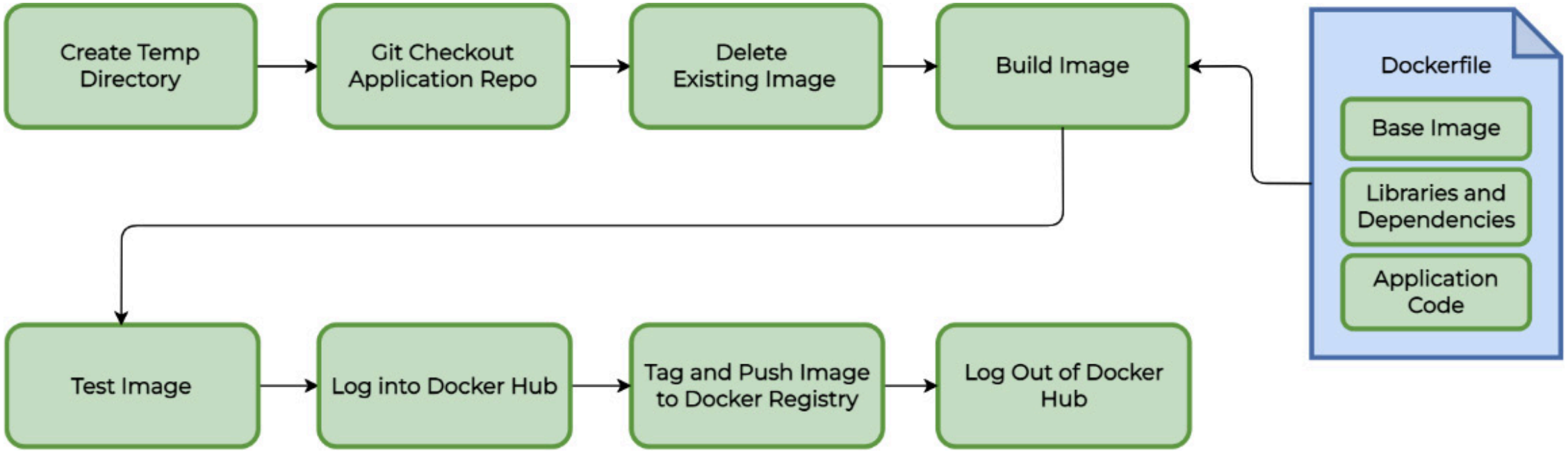
> L'action stop, stop et remove le CT

```
***********
ok: [rocky-docker]
```

Docker

Docker build

> Tentons decréer le pipeline suivant avec Ansible



Docker build

```
$ mkdir vars
```

- On créé le credential sur `Dockerhub` (Account Settings/Security)
- On crée ensuite un un fichier `vault` pour stocker le secret de connexion

```
ansible-vault create vars/docker-credential.yaml
```

```
docker_username: ansible
docker_password: dckr_pat_~~~aYzktJFgY
```

> Le mien : https://hub.docker.com/repository/docker/alexdocker6lexorg/

Docker build

> Regardons le playbook qui permet de créer le pipeline : Pipeline-Docker

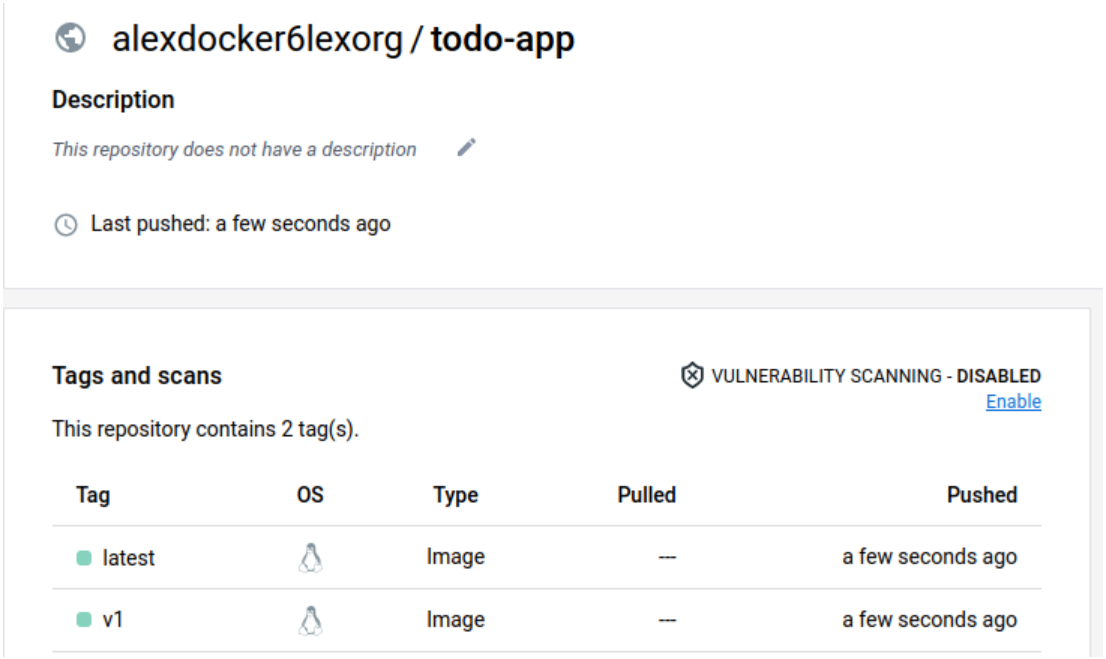
```
- name: Building Container Images
  hosts: "{{ NODES }}"
  become: yes
  vars:
    application_repo: https://github.com/ginigangadharan/nodejs-todo-demo-app
    application_branch: main
    application_name: todo-app
    application_version: v1
    container_image_repository: ginigangadharan
    container_registry_url: https://index.docker.io/v1/

  vars_files:
    - vars/docker-credential.yaml
  tasks:
    - name: Create temporary location
      ansible.builtin.tempfile:
        state: directory
        prefix: "container_build_"
      register: temp_location
[...]
```

Docker build

- Lançons le playbook

```
$ ansible-playbook container-build.yaml -e "NODES=dockerhost" --ask-vault-password
```



Docker Run container

- On lance le playbook `container-manage.yaml` utilisé précédemment pour lancer le CT sur la cible

```
$ ansible-playbook container-manage.yaml \
    -e "NODES=dockerhost \
    container_image=alexdocker6lexorg/todo-app \
    container_name=todo-app \
    container_port=3000 container_expose_port=8081"
```

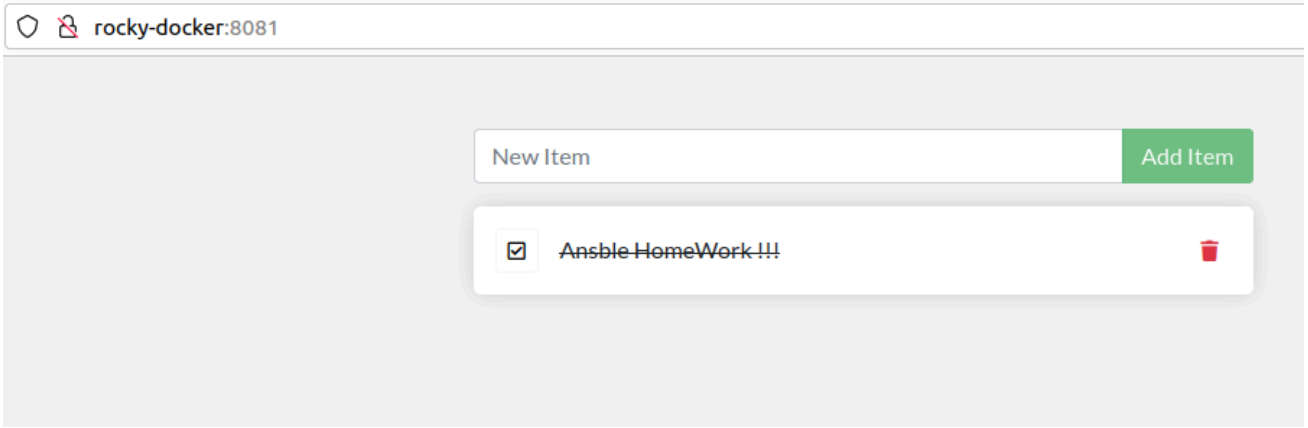
```
PLAY [Manage Docker containers]
[...]
rocky-docker               : ok=4    changed=1
```

Docker Run container

- Sur la cible

```
# docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS                    NAMES
f966098f4e51   alexdocker6lexorg/todo-app   "docker-entrypoint.s…"   13 seconds ago   Up 12 seconds   0.0.0.0:8081->3000/tcp   todo-app
```

- Le résultat



Docker Stop container

- On reprend le playbook `container-manage.yaml` en ajoutant l'option `container_action=stop`

```
$ ansible-playbook container-manage.yaml \
    -e "NODES=dockerhost \
    container_image=alexdocker6lexorg/todo-app \
    container_name=todo-app \
    container_port=3000 \
    container_expose_port=8081 \
    container_action=stop"
```

- Ce qui donne sur la cible

```
# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

Managing multi-container applications

> Nous allons créer un playbook pour déployer Wordpress sur 2 containers

- Le playbook `deploy-wordpress-on-docker.yaml`

```
---
- name: Deploy wordpress stack on Docker
  hosts: "{{ NODES }}"
  become: yes
  vars:
    db_volume: 'mariadb'
    wordpress: 'wordpress'
    mysql_root_password: 'secretrootpassword'
    mysql_username: 'wordpressuser'
    mysql_password: 'secretpassword'
    mysql_database: 'wordpressdb'
    container_port: 8082
```

```
tasks:
  - name: Deploy MariaDB server for Database
    community.docker.docker_container:
      state: started
      image: mariadb
      name: mariadb
      volumes:
        - "{{db_volume}}:/var/lib/mysql"
      env:
        MYSQL_ROOT_PASSWORD: "{{ mysql_root_password }}"
        MYSQL_PASSWORD: "{{ mysql_password }}"
        MYSQL_DATABASE: "{{ mysql_database }}"
        MYSQL_USER: "{{ mysql_username }}"
```

```
  - name: Deploy WordPress
    community.docker.docker_container:
      state: started
      image: wordpress
      name: wordpress
      restart_policy: always
      ports:
        - "{{ container_port }}:80"
      links:
        - "{{ db_volume }}:/var/lib/mysql"
      volumes:
        - "{{ wordpress }}:/var/www/html"
      env:
        MYSQL_PASSWORD: "{{ mysql_password }}"
        MYSQL_DATABASE: "{{ mysql_database }}"
        MYSQL_USER: "{{ mysql_username }}"
        MYSQL_HOST: mariadb
```

Managing multi-container applications

- On lance le playbook sur la cible

```
$ ansible-playbook deploy-wordpress-on-docker.yaml -e "NODES=dockerhost"
```

```
TASK [Gathering Facts] *********************************************************
ok: [rocky-docker]

TASK [Deploy MariaDB server for Database] **************************************
changed: [rocky-docker]

TASK [Deploy WordPress] ********************************************************
changed: [rocky-docker]

PLAY RECAP *********************************************************************
rocky-docker               : ok=3    changed=2
```

Managing multi-container applications

- Sur la cible

```
# docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED         STATUS         PORTS                    NAMES
ef3cdea34bb4   wordpress   "docker-entrypoint.s…"   3 minutes ago   Up 3 minutes   0.0.0.0:8082->80/tcp   wordpress
59d587ada613   mariadb     "docker-entrypoint.s…"   3 minutes ago   Up 3 minutes   3306/tcp               mariadb
```