Projet 1 : Intégration Continue

Cahier des charges

PROJET FONDAMENTAUX Mise en place d'une chaîne d'intégration continue avec AWS.

Première phase : Installation automatisée d'un environnement de Labs via ansible.

- Une VM en local type Linux. (VirtualBox, WSL, GitBash, SSH).
- Une VM serveur Web (apache2).
- Une VM GitLab.

Seconde phase : Mise en place via GitLab du CI (Continuous Integration) pour la démonstration nous cherchons à déployer de façon automatisée l'intégration de pages web statiques. Le framework utilisé sera JEKYLL (à vous de le découvrir). Faire un test de Jekyll avec un exemple de site (il existe de nombreux exemples). Nous aimerions un déclenchement de l'intégration des pages web (Jekyll) sur la machine virtuelle serveur web AWS.

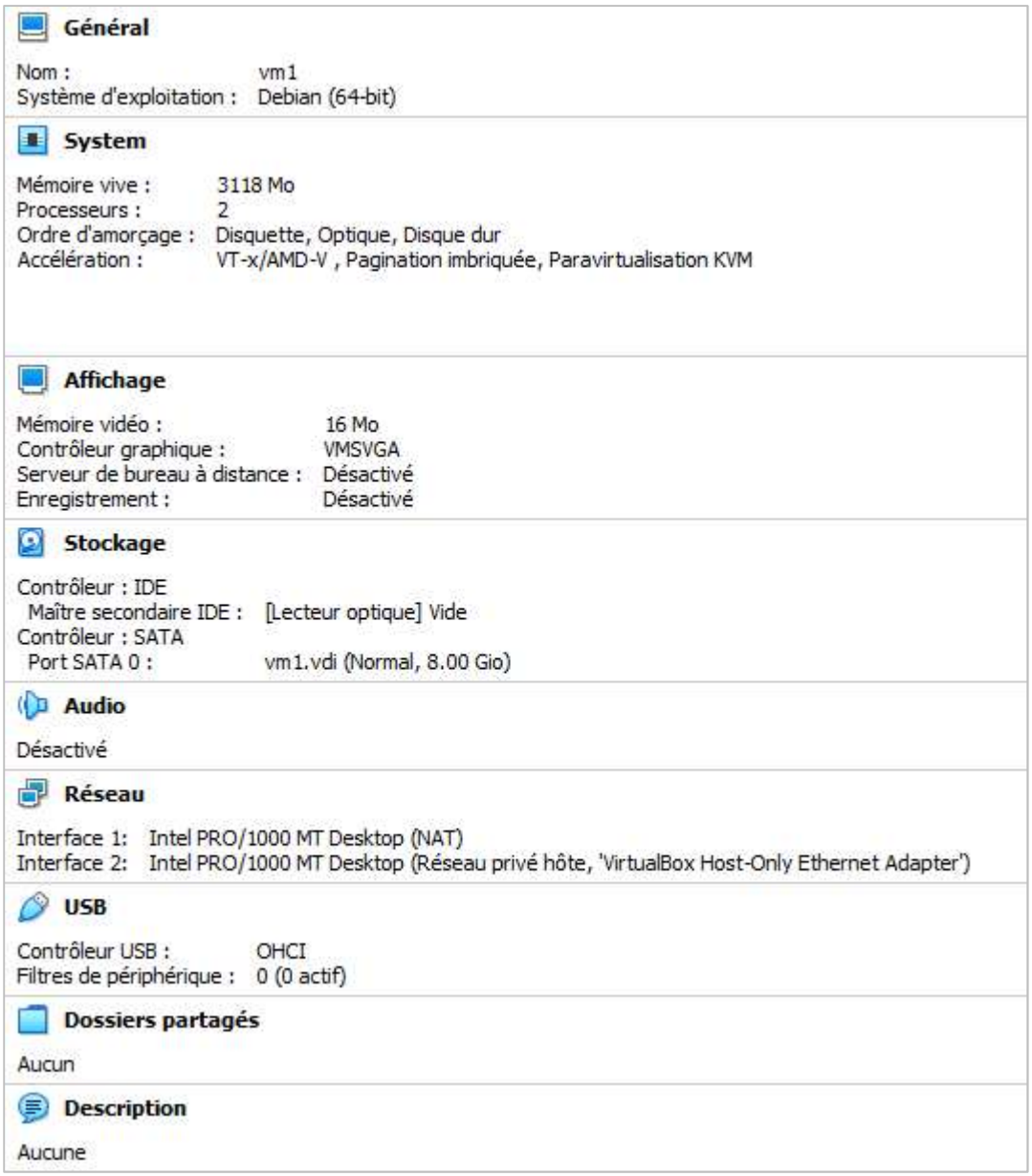
GitHub

https://github.com/mmmats/projet-ci.git

Réalisation

Première phase : création des VM

Création manuelle de la VM locale avec VirtualBox (vm1)



Installation ansible sur vm1

Création d'un user AWS pour l'obtention d'une clé de connexion :

Compte > Mes identifiants de sécurité > Utilisateurs > Ajouter un utilisateur

Penser à cocher le groupe de sécurité à rejoindre disposant des droits suffisant pour réaliser les actions demandées (création d'instance ec2, vpc etc…)

Création des vm2 (webserver) et vm3 (webserver) via ansible playbook :

```
ansible-playbook playbook.yml
```

playbook.yml

```yaml
---

- name: projet1
  hosts: localhost
  vars:
    aws_access_key: "myaccesskey"
    aws_secret_key: "mysecretkey"
    region: "eu-central-1"

    # VPC
    vpc_cidr: 10.10.0.0/24
    vpc_name: "VPC mats"

    # Subnet
    subnet_name: "Subnet mats"
    subnet_cidr: 10.10.0.0/26

    # Internet Gateway Name
    igw_name: "GW mats"

    securitygroup_name: "SecurityGroup mats"

    ec2_tag: "WebServer mats"

    #The local path to which we would save our EC2 Private Key
    ec2_key_directory: "/home/ansible/roles/aws-vpc/"
    keypair_name: "ec2_key_pair"

  tasks:
    - name: create VPC
      ec2_vpc_net:
        name: "VPC mats"
        cidr_block: "{{ vpc_cidr }}"
        region: "{{ region }}"
        state: present
        aws_access_key: "{{ aws_access_key }}"
        aws_secret_key: "{{ aws_secret_key }}"
      register: vpc

    - name: associate subnet to the VPC
      ec2_vpc_subnet:
       state: present
       vpc_id: "{{ vpc.vpc.id }}"
       region: "{{ region }}"
       cidr: "{{ subnet_cidr }}"
       aws_access_key: "{{ aws_access_key }}"
       aws_secret_key: "{{ aws_secret_key }}"
       map_public: yes
       resource_tags:
         Name: "{{ subnet_name }}"
      register: subnet

    - name: create IGW
      ec2_vpc_igw:
       vpc_id: "{{ vpc.vpc.id }}"
       region: "{{ region }}"
       aws_access_key: "{{ aws_access_key }}"
       aws_secret_key: "{{ aws_secret_key }}"
       state: "present"
       tags:
         Name: "{{ igw_name }}"
      register: igw

    - name: Route IGW
      ec2_vpc_route_table:
        vpc_id: "{{ vpc.vpc.id }}"
        region: "{{ region }}"
        aws_access_key: "{{ aws_access_key }}"
        aws_secret_key: "{{ aws_secret_key }}"
        subnets:
          - "{{ subnet.subnet.id }}"
        routes:
          - dest: 0.0.0.0/0
            gateway_id: "{{ igw.gateway_id  }}"
```

```
    # update the CIDR address in the SSH port section.

    - name: Create Security Group
      ec2_group:
        name: Web DMZ
        description: DMZ Security Group
        vpc_id: "{{ vpc.vpc.id }}"
        region: "{{ region }}"
        aws_access_key: "{{ aws_access_key }}"
        aws_secret_key: "{{ aws_secret_key }}"
        rules:
          - proto: tcp
            ports:
            - 80
            cidr_ip: 0.0.0.0/0
          - proto: tcp
            ports:
            - 22
            cidr_ip: 0.0.0.0/0
      register: security_group

    - name: create a new ec2 key pair
      ec2_key:
        aws_access_key: "{{ aws_access_key }}"
        aws_secret_key: "{{ aws_secret_key }}"
        name: ec2_keypair
        region: "{{ region }}"
      register: keypair

    - name: Copy EC2 Private Key locally so it can be later on used to SSH into the instance
      copy: content="{{ keypair.key.private_key }}" dest={{ ec2_key_directory }}key.ppk
      when: keypair.changed == true

    - name: Create EC2 web server
      ec2:
        image: ami-0f1026b68319bad6c
        wait: yes
        instance_type: t2.micro
        region: "{{ region }}"
        group_id: "{{ security_group.group_id }}"
        vpc_subnet_id: "{{ subnet.subnet.id }}"
        key_name: "{{ keypair.key.name  }}"
        count_tag:
          Name: apacheserver
        exact_count: 1
        aws_access_key: "{{ aws_access_key }}"
        aws_secret_key: "{{ aws_secret_key }}"
      register: apachemats

    - name: Create EC2 git server
      ec2:
        image: ami-0f1026b68319bad6c
        wait: yes
        instance_type: t2.large
        region: "{{ region }}"
        group_id: "{{ security_group.group_id }}"
        vpc_subnet_id: "{{ subnet.subnet.id }}"
        key_name: "{{ keypair.key.name  }}"
        count_tag:
          Name: gitserver
        exact_count: 1
        aws_access_key: "{{ aws_access_key }}"
        aws_secret_key: "{{ aws_secret_key }}"
      register: gitmats

...
```

Ajout des IP vm2 et vm3 dans vm1:

/etc/ansible/hosts

```
[webservers]
18.156.6.212

[gitservers]
18.184.13.162
```

Installation et lancement du container httpd sur vm2 via ansible playbook :

```
ansible-playbook configVM.yml
```

configVM.yml

```
---

- name: Installation webservers
  hosts: webservers
  remote_user: admin
  become: yes
  vars:
    ansible_ssh_private_key_file: "/home/ansible/roles/aws-vpc/key.ppk"
  tasks:
    - name: Installation required packages
      apt:
        name: "{{item}}"
        state: latest
        update_cache: yes
      loop:
        - apt-transport-https
        - ca-certificates
        - curl
        - gnupg
        - lsb-release
    - name: Docker apt key
      apt_key:
        url: https://download.docker.com/linux/debian/gpg
        state: present
    - name: Docker apt repository
      apt_repository:
        repo: deb https://download.docker.com/linux/debian buster stable
        state: present
    - name: Installation Docker
      apt:
        name: "{{item}}"
        state: latest
        update_cache: yes
      loop:
        - docker-ce
        - docker-ce-cli
        - containerd.io
    - name: Installation python-docker
      apt:
        name: python-docker
    - name: Lancement container apache2
      docker_container:
        name: apache2
        image: httpd
        state: started
        restart: yes
        recreate: yes
        ports:
          - "80:80"
        volumes:
          - "/var/www/html/:/usr/local/apache2/htdocs/"

...
```

Changement du port ssh de la vm3 dans /etc/ssh/sshd_config (port : 22000 car nous utiliserons le port 22 pour gitlab)

Installation manuelle de docker sur vm3 (gitlab) :

- docker

```
sudo apt-get update

sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

echo \
  "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io
```

- docker-compose

```
apt install docker-compose
```

Lancement du container GitLab via docker-compose sur vm3 (gitlab)

Ajout de la variable d'environnement $GITLAB_HOME

```
export GITLAB_HOME=/srv/gitlab
```

Lancement du docker-compose.yml

```
docker-compose up -d
```

docker-compose.yml

```
---
web:
  image: 'gitlab/gitlab-ce:latest'
 # restart: always
  hostname: 'mats-gitlab'
  environment:
    GITLAB_OMNIBUS_CONFIG: |
      external_url 'http://18.184.13.162'
      # Add any other gitlab.rb configuration here, each on its own line
  ports:
    - '80:80'
    - '443:443'
    - '22:22'
  volumes:
    - '$GITLAB_HOME/config:/etc/gitlab'
    - '$GITLAB_HOME/logs:/var/log/gitlab'
    - '$GITLAB_HOME/data:/var/opt/gitlab'
...
```

Seconde phase : Mise en place de l'intégration continue

Connexion sur GitLab : http://18.184.13.162/

Initialisation du compte root/password

Création d'un projet « **website_auto_update** »

Clonage du projet sur vm1 (localhost)

Création du gitlab-runner et association avec le projet gitlab

```
# Download the binary for your system
sudo curl -L --output /usr/local/bin/gitlab-runner https://gitlab-runner-
downloads.s3.amazonaws.com/latest/binaries/gitlab-runner-linux-amd64

# Give it permissions to execute
sudo chmod +x /usr/local/bin/gitlab-runner

# Create a GitLab CI user
sudo useradd --comment 'GitLab Runner' --create-home gitlab-runner --shell /bin/bash

# Install and run as service
sudo gitlab-runner install --user=gitlab-runner --working-directory=/home/gitlab-runner
sudo gitlab-runner start
sudo gitlab-runner register --url http://18.184.13.162/ --registration-token 5ayByArcvudxqsMsAJyL
```

Téléchargement du package jekyll sur vm1 (localhost)

Création d'un site à partir d'un template jekyll

```
jekyll new /home/mats/projet1/website_auto_update
```

Git commit & push

```
git add .
git commit -m "site creation with jekyll"
git push origin master
```

Autoriser la connexion SSH via password sur vm2 (webserver)

/etc/ssh/sshd_config

```
PasswordAuthentication yes
```

Initialiser le password pour le compte « admin » sur vm2

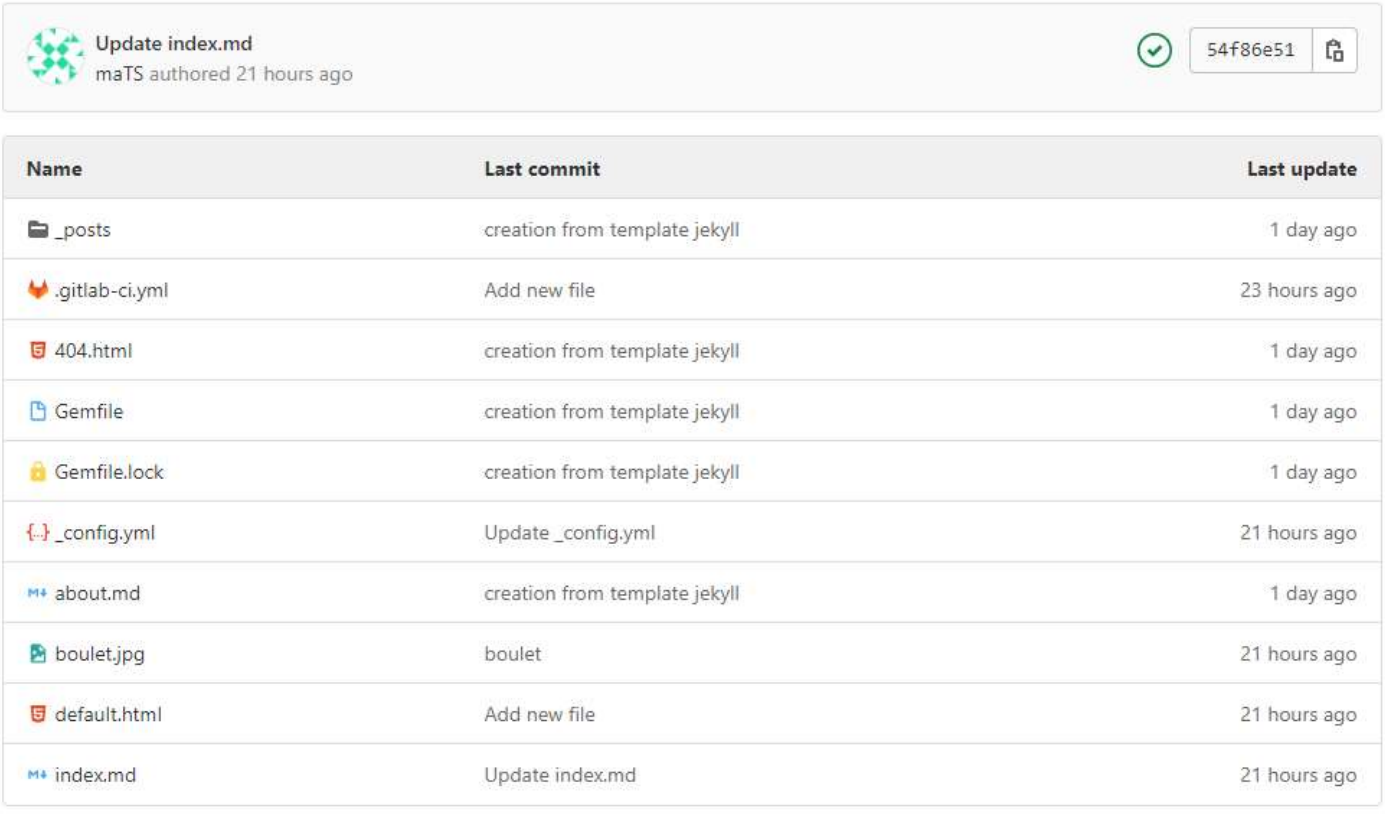
```
sudo passwd admin
```

Création du fichier .gitlab-ci.yml pour paramétrer l'intégration continue

.gitlab-ci.yml

```
---
# This file is a template, and might need editing before it works on your project.
# Template project: https://gitlab.com/pages/jekyll
# Docs: https://docs.gitlab.com/ce/pages/
image: ruby:2.6
before_script:
- bundle install
- apt-get update -qq && apt-get install -y -qq sshpass
stage_deploy:
  artifacts:
    paths:
    - build/
  only:
  - master
  script:
  - bundle exec jekyll build -d build/
  - sshpass -V
  - export SSHPASS=$USER_PASS
  - cd build/
  - sshpass -e scp -o stricthostkeychecking=no -
r /builds/root/website_auto_update/build/* admin@18.156.6.212:/var/www/html
...
```

GitLab project: **website_auto_update**



Intégration continue déployée – exemple de job réalisé avec succès