TP - AUTOMATISATION DU DÉPLOIEMENT D'UNE INFRASTRUCTURE AVEC ANSIBLE

05/03/2025

I) D'abord, je commence par crée 2 machine virtuelles sur Microsoft Azure (VM1 et VM2)

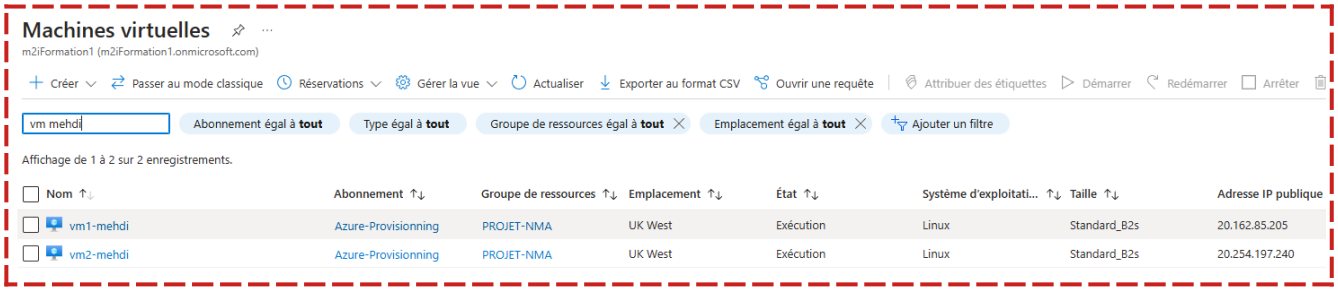
Pour rappel / prérequis:

MobaXterm pour se connecter aux machines en SSH grâce aux clés RSA

VM1 (10.0.0.6) : Héberge **Jenkins, Git et Ansible**.

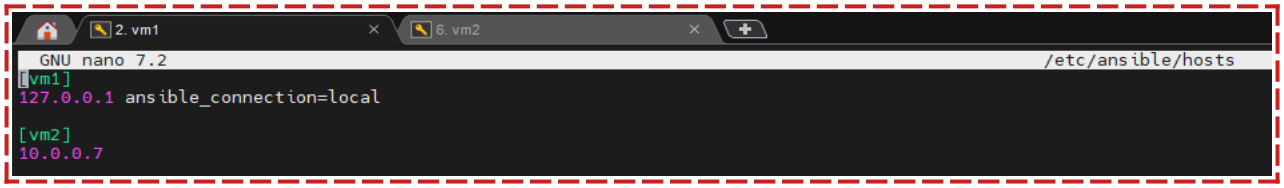
VM2 (10.0.0.7) : Héberge **MySQL, Docker et Nginx**.

Ouvrir les ports nécessaires (**22, 80, 443, 8080**) dans le groupe de sécurité réseau.



II) Sur **VM1**, installer **Ansible** et créer un fichier `inventory` pour référencer les VMs :

sudo apt update && sudo apt install -y ansible



III) Je peux tester les connexions entre mes machines

```
azureuser@vm1-mehdi:~$ ansible -i /etc/ansible/hosts all -m ping
127.0.0.1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
10.0.0.7 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

IV) Déploiement Automatisé avec Ansible

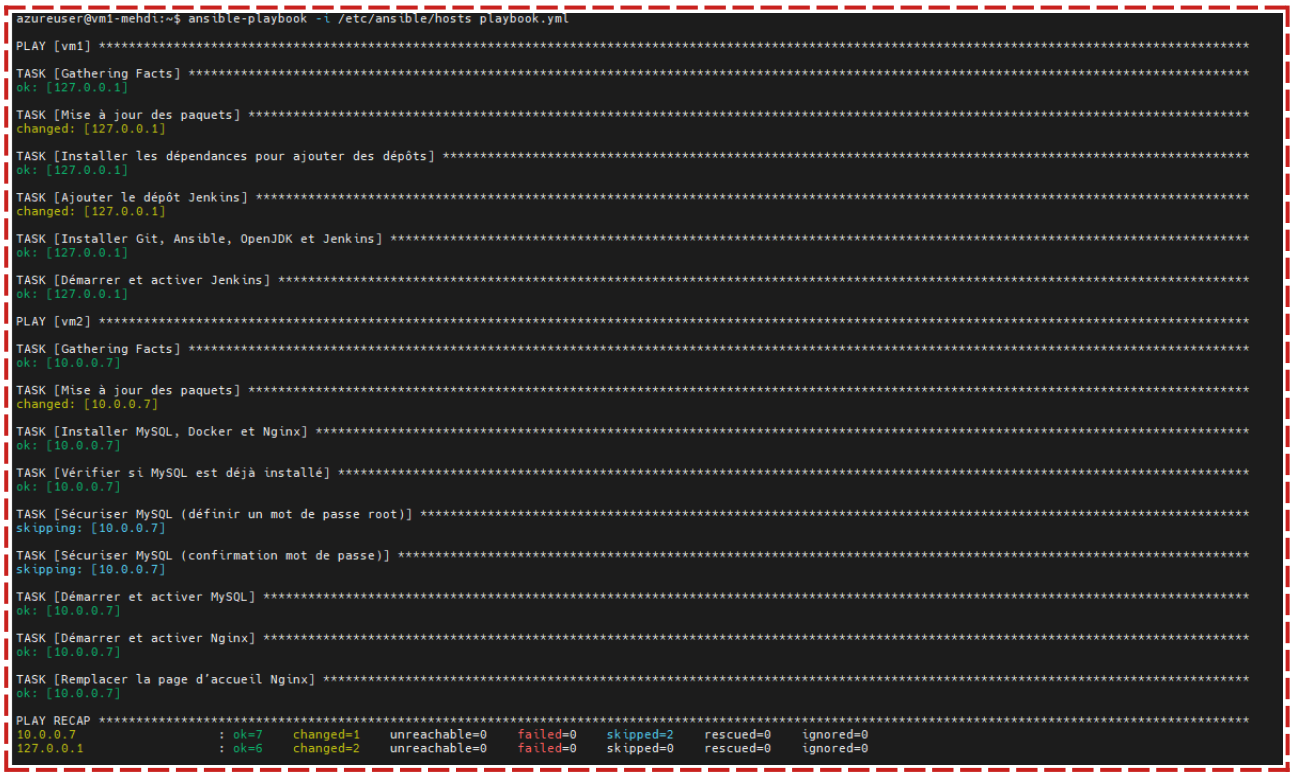
Création d'un fichier **`playbook.yml`** qui :

- Installe **Jenkins, Git et Ansible** sur **VM1**.
- Installe **MySQL, Docker et Nginx** sur **VM2**.
- Sécurise **MySQL** et configure **Nginx**.

```
azureuser@vm1-mehdi:~$ cat playbook.yml
- hosts: vm1
  become: yes
  tasks:
    - name: Mise à jour des paquets
      apt:
        update_cache: yes

    - name: Installer les dépendances pour ajouter des dépôts
      apt:
        name:
          - curl
          - gnupg
          - software-properties-common
        state: present

    - name: Ajouter le dépôt Jenkins
      shell: |
        curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | tee \
          /usr/share/keyrings/jenkins-keyring.asc > /dev/null
        echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
          https://pkg.jenkins.io/debian binary/" | tee \
          /etc/apt/sources.list.d/jenkins.list > /dev/null
        apt update

    - name: Installer Git, Ansible, OpenJDK et Jenkins
      apt:
        name:
          - git
          - ansible
          - openjdk-17-jdk
          - jenkins
        state: present

    - name: Démarrer et activer Jenkins
      service:
        name: jenkins
        state: started
        enabled: yes

- hosts: vm2
  become: yes
  tasks:
    - name: Mise à jour des paquets
      apt:
        update_cache: yes

    - name: Installer MySQL, Docker et Nginx
      apt:
        name:
          - mysql-server
          - docker.io
          - nginx
        state: present

    - name: Vérifier si MySQL est déjà installé
      stat:
        path: /etc/mysql/my.cnf
      register: mysql_installed

    - name: Sécuriser MySQL (définir un mot de passe root)
      debconf:
        name: mysql-server
        question: "mysql-server/root_password"
        value: "rootpassword"
        vtype: "password"
      when: mysql_installed.stat.exists == False

    - name: Sécuriser MySQL (confirmation mot de passe)
      debconf:
        name: mysql-server
        question: "mysql-server/root_password_again"
        value: "rootpassword"
        vtype: "password"
      when: mysql_installed.stat.exists == False

    - name: Démarrer et activer MySQL
      service:
        name: mysql
        state: started
        enabled: yes

    - name: Démarrer et activer Nginx
      service:
        name: nginx
        state: started
        enabled: yes

    - name: Remplacer la page d'accueil Nginx
      copy:
        content: "Bienvenue sur le serveur web de Mehdi pour le TP AUTOMATISATION DU DÉPLOIEMENT D'UNE INFRASTRUCTURE AVEC ANSIBLE !"
        dest: /var/www/html/index.html
```

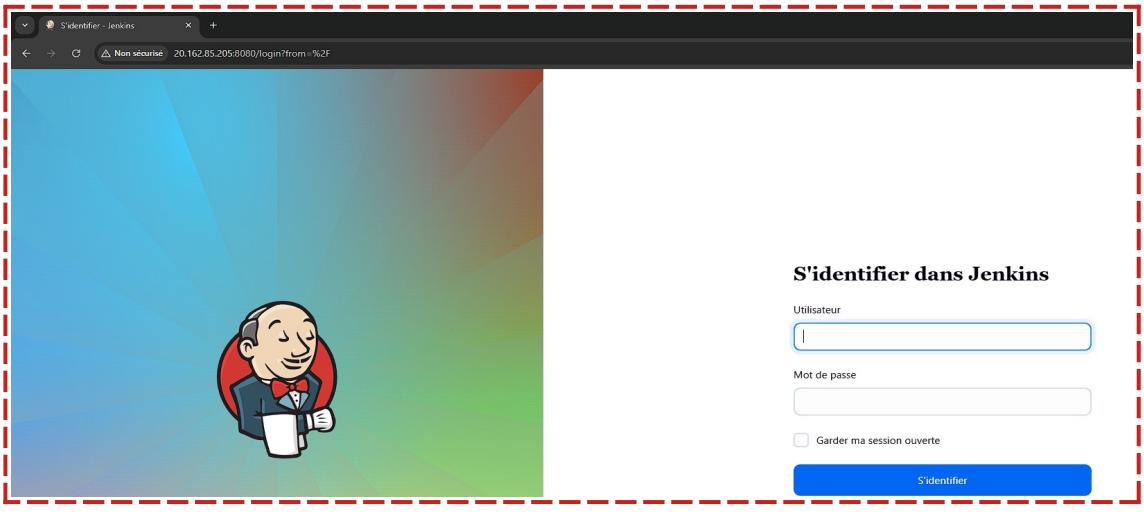
V) Exécution du playbook.yml + vérification du bon fonctionnement

azureuser@vm1-mehdi:~$ ansible-playbook -i /etc/ansible/hosts playbook.yml



Sur un navigateur avec l'ip Publique de ma VM2 :

Sur un navigateur avec l'ip Publique de ma VM1 au port 8080 :



Vérification de la bonne connexion de MySQL sur la VM2 :

```
azureuser@vm2-mehdi:~$ sudo mysql -u root -p -e "SHOW DATABASES;"
Enter password:
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
```

VI) Ajout de la récupération du projet via Github sur le playbook.yml

```
    - name: Cloner le projet depuis GitHub
      git:
        repo: "https://github.com/Meeeehddiiii/simple_serveur_web.git"
        dest: "/home/azureuser/simple_serveur_web"
        version: main
```

VII) **Création du Dockerfile :** Un **Dockerfile** a été créé pour construire l'image Docker avec Nginx comme serveur web et le projet cloné comme contenu

```
    - name: Créer le Dockerfile
      copy:
        dest: "/home/azureuser/simple_serveur_web/Dockerfile"
        content: |
          FROM nginx:latest
          COPY . /usr/share/nginx/html
          EXPOSE 80
          CMD ["nginx", "-g", "daemon off;"]
```

VIII) Création de l’image docker + supprimer si il existe un ancien conteneur nommée simple_serveur_weba

```
    - name: Construire l'image Docker
      shell: |
        cd /home/azureuser/simple_serveur_web
        docker build -t simple_serveur_web .

    - name: Supprimer l'ancien conteneur s'il existe
      shell: docker rm -f simple_serveur_web || true
```

IX) Exécution du conteneur Docker sur le port 8081

```
  - name: Exécuter le conteneur Docker
    shell: |
      docker run -d -p 8081:80 --name simple_serveur_web simple_serveur_web
```

X) Création du fichier « credentials.yml » contenant les identifiants (username + token) Docker Hub

```
 GNU nano 7.2                                                credentials.yml *
username: "meeeehddiiii"
password: "dckr_pat_N4*Xlw9*Nl2m*B*4v*h*hd*Q*8w"
```

XI) Chiffrer le fichier via la commande :

sudo ansible-vault encrypt credentials.yml

```
azureuser@vm1-mehdi:~$ sudo ansible-vault encrypt credentials.yml
New Vault password:
Confirm New Vault password:
Encryption successful
```

XII) Se connecter à Docker Hub + Taguer l'image + Pousser l'image sur Docker Hub

```
- name: Se connecter à Docker Hub
  shell: echo "{{ docker_hub_token }}" | docker login -u "{{ docker_hub_username }}" --password-stdin

- name: Taguer l'image Docker pour Docker Hub
  shell: docker tag simple_serveur_web {{ docker_hub_username }}/simple_serveur_web:latest

- name: Pousser l'image Docker sur Docker Hub
  shell: docker push {{ docker_hub_username }}/simple_serveur_web:latest
```

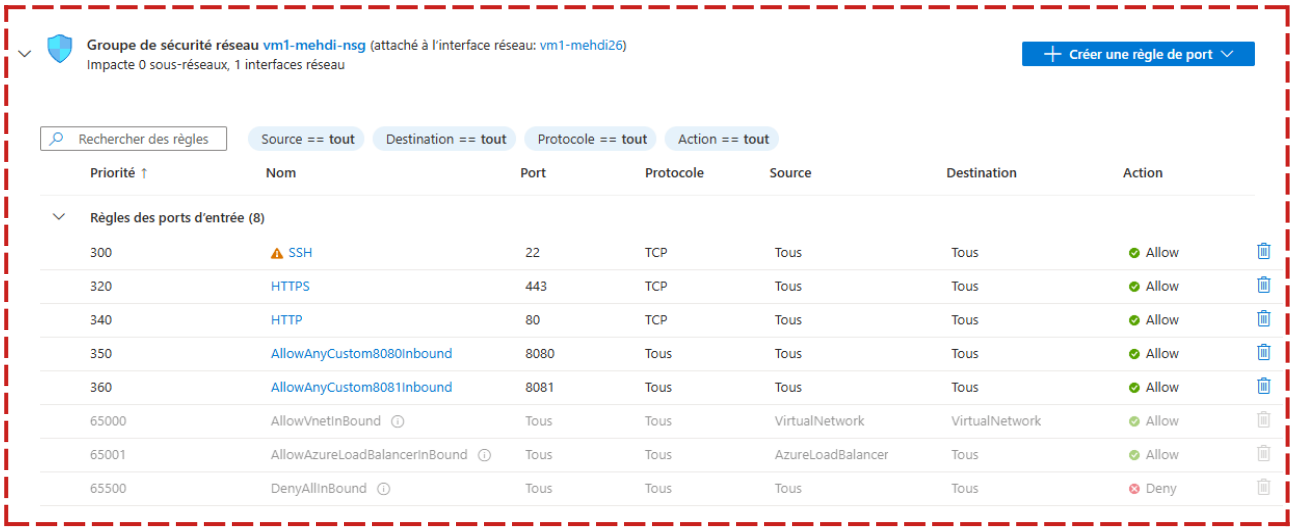
Le fichier « playbook.yml » est disponible ici : cliquez_ici

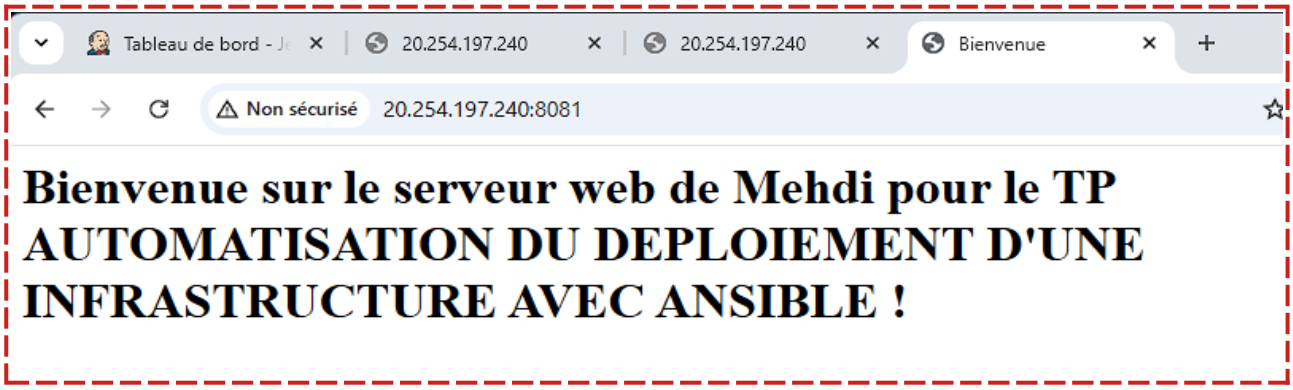
XIII) On peut désormais lancer le playbook.yml grâce à la commande suivante

sudo ansible-playbook playbook.yml –ask-vault-pass

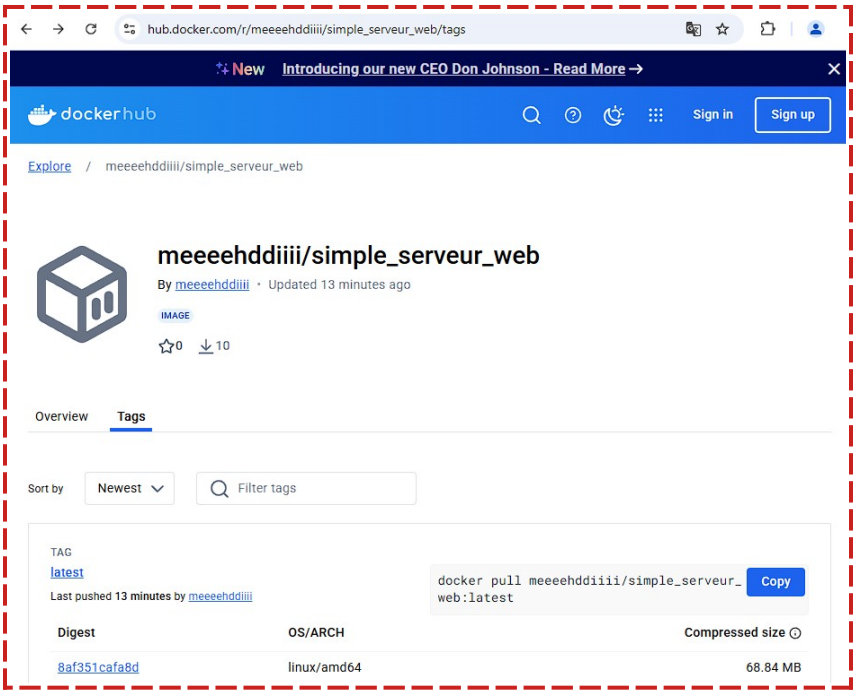
```
PLAY [vm1] *********************************************************************

TASK [Gathering Facts] *********************************************************
ok: [127.0.0.1]

TASK [Mise à jour des paquets] *************************************************
changed: [127.0.0.1]

TASK [Installer les dépendances pour ajouter des dépôts] ***********************
ok: [127.0.0.1]

TASK [Ajouter le dépôt Jenkins] ************************************************
changed: [127.0.0.1]

TASK [Installer Git, Ansible, OpenJDK et Jenkins] ******************************
ok: [127.0.0.1]

TASK [Démarrer et activer Jenkins] *********************************************
ok: [127.0.0.1]

PLAY [vm2] *********************************************************************

TASK [Gathering Facts] *********************************************************
ok: [10.0.0.7]

TASK [Mise à jour des paquets] *************************************************
changed: [10.0.0.7]

TASK [Installer MySQL, Docker et Nginx] ****************************************
ok: [10.0.0.7]

TASK [Vérifier si MySQL est déjà installé] *************************************
ok: [10.0.0.7]

TASK [Sécuriser MySQL (définir un mot de passe root)] **************************
skipping: [10.0.0.7]

TASK [Sécuriser MySQL (confirmation mot de passe)] *****************************
skipping: [10.0.0.7]

TASK [Démarrer et activer MySQL] ***********************************************
ok: [10.0.0.7]

TASK [Démarrer et activer Nginx] ***********************************************
ok: [10.0.0.7]

TASK [Cloner le projet depuis GitHub] ******************************************
ok: [10.0.0.7]

TASK [Créer le Dockerfile] *****************************************************
ok: [10.0.0.7]

TASK [Construire l'image Docker] ***********************************************
changed: [10.0.0.7]

TASK [Supprimer l'ancien conteneur s'il existe] ********************************
changed: [10.0.0.7]

TASK [Exécuter le conteneur Docker] ********************************************
changed: [10.0.0.7]

TASK [Se connecter à Docker Hub] ***********************************************
changed: [10.0.0.7]

TASK [Taguer l'image Docker pour Docker Hub] ***********************************
changed: [10.0.0.7]

TASK [Pousser l'image Docker sur Docker Hub] ***********************************
changed: [10.0.0.7]

PLAY RECAP *********************************************************************
10.0.0.7                   : ok=14   changed=7    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
127.0.0.1                  : ok=6    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

XIV) Désormais, j'ajoute une règle de **ports entrants** sur le **groupe de sécurité réseau (NSG)** dans Azure autorisant le trafic vers le port 8081 afin de pouvoir y accéder depuis un navigateur





XIV) Vérification que le push vers Docker Hub a bien fonctionné directement sur le site de Docker Hub.



XV) Configuration Pipeline Jenkins | Création du Jenkinsfile

```
 GNU nano 7.2                                                    Jenkinsfile
pipeline {
    agent any

    stages {
        stage('Checkout') {
            steps {
                // Cloner le dépôt GitHub en spécifiant la branche "main"
                git branch: 'main', url: 'https://github.com/Meeeehddiiii/simple_serveur_web.git'
            }
        }

        stage('Vérification de la syntaxe HTML') {
            steps {
                // Vérification de la syntaxe du fichier index.html avec suppression de l'échec en cas d'avertissement
                script {
                    def status = sh(script: 'tidy -e index.html', returnStatus: true)
                    if (status != 0) {
                        echo "Des avertissements ont été trouvés dans le fichier HTML."
                    } else {
                        echo "Aucun problème de syntaxe trouvé dans le fichier HTML."
                    }
                }
            }
        }
    }
}
```

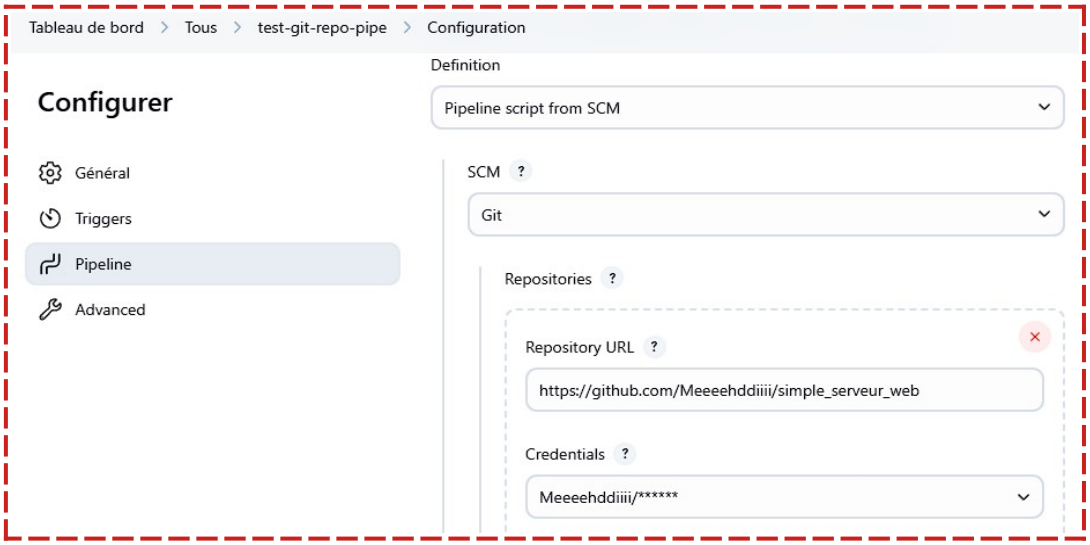
Le Jenkinsfile que nous avons mis en place sert à automatiser un processus de validation de la syntaxe HTML de ton fichier index.html sur chaque commit poussé dans le dépôt GitHub.

Il permet de détecter les erreurs majeures et mineures dans la syntaxe du fichier HTML en utilisant l'outil tidy.

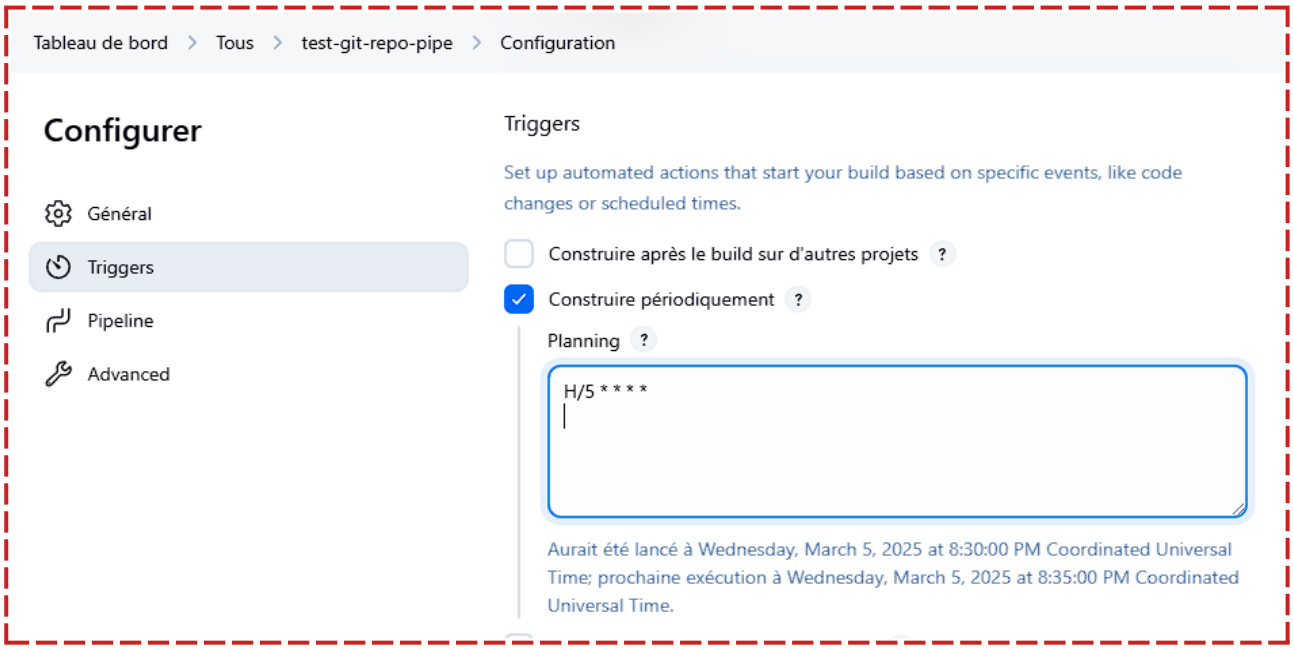
En résumé :

- **Les erreurs graves (majeures)**, comme les erreurs de balises ou d'autres violations importantes de la syntaxe HTML, seront détectées et rapportées.
- **Les avertissements (mineurs)**, comme l'absence de la déclaration `<!DOCTYPE>`, seront également signalés, mais ne feront pas échouer le pipeline à moins que ce comportement ne soit configuré autrement.

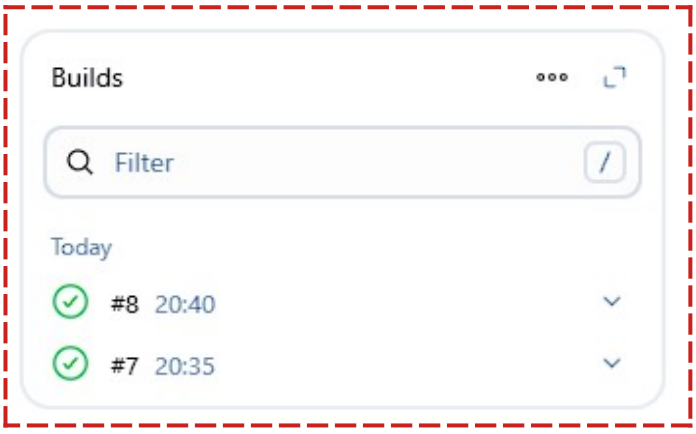
XVI) Configuration Pipeline Jenkins | Création de l'item



Je configure un trigger afin qu'un build soit créé toutes les 5 minutes.



Désormais, je peux apercevoir qu’un build est crée toutes les 5 minutes



Fiche 4: Préparer un environnement de test

06/03/2025

I) Configuration de l'Agent Jenkins

Pour rappel / prérequis:

Pour permettre à **VM1** (serveur Jenkins) de se connecter à **VM2** (l'agent), une connexion SSH sécurisée a été établie entre les deux machines.

Objectif :

Mettre en place un agent Jenkins sur la machine virtuelle **VM2**, en utilisant **VM1** (le serveur Jenkins) pour gérer les tâches à distance.

II) Installation et Préparation sur VM2 (Agent Jenkins)

Avant de configurer l'agent Jenkins, il fallait s’assurer que **VM2** était prête pour recevoir les tâches Jenkins. Nécessitant Java pour son fonctionnement, J’installe **OpenJDK 11** :

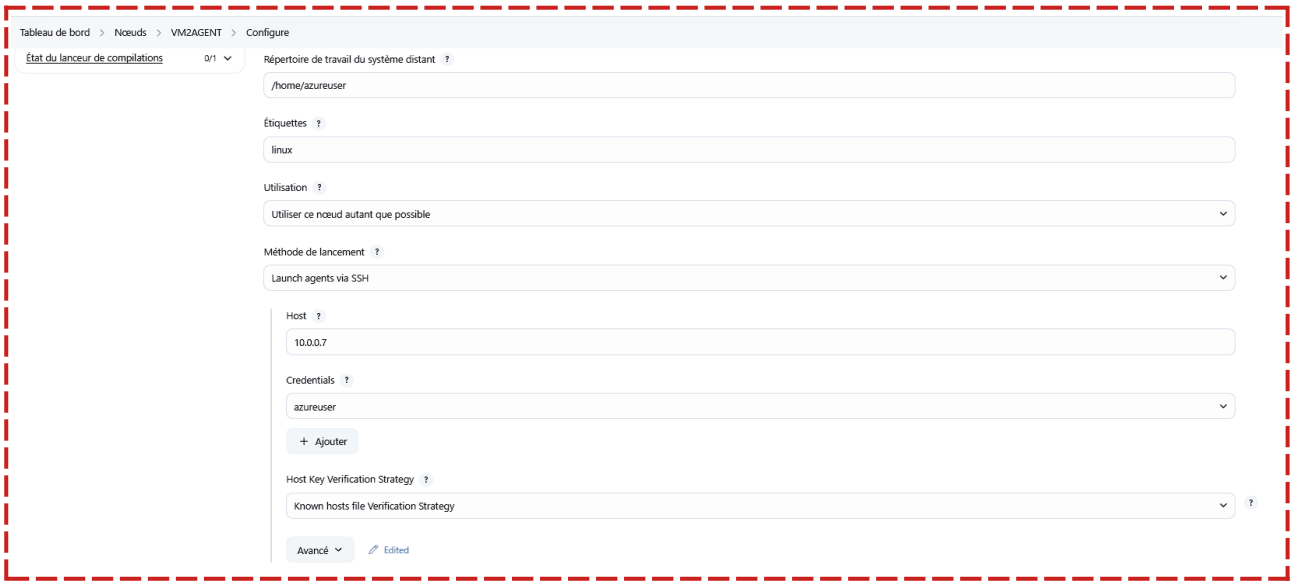
sudo apt update
sudo apt install openjdk-11-jdk

Vérifier que Java est bien installer :

```
azureuser@vm2-mehdi:~$ java --version
openjdk 21.0.6 2025-01-21
OpenJDK Runtime Environment (build 21.0.6+7-Ubuntu-124.04.1)
OpenJDK 64-Bit Server VM (build 21.0.6+7-Ubuntu-124.04.1, mixed mode, sharing)
```

II) Configuration de l'Agent Jenkins

Depuis l'interface web de Jenkins sur **VM1**, un nouvel agent a été ajouté dans **"Manage Jenkins" > "Manage Nodes and Clouds"**

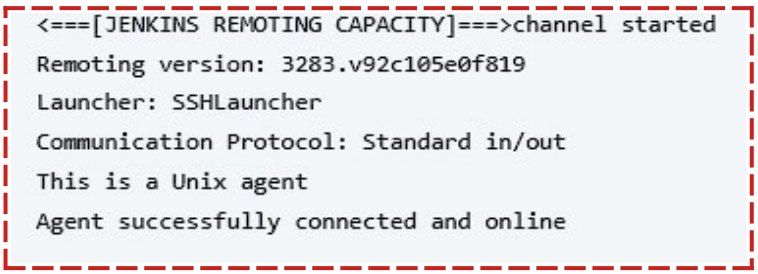


Résumé de la Configuration de l'Agent Jenkins :

- **Nom de l'Agent** : `VM2Agent`
- **Méthode de Lancement** : `Launch agent by SSH`
- **Host** : `10.0.0.7` (IP de l'agent **VM2**)
- **Credentials** : **Username** : `azureuser` (utilisateur sur **VM2**) + clé SSH permettre l'authentification SSH entre **VM1** et **VM2**
- **Répertoire de travail du système distant** : `/home/azureuser` (répertoire d'accueil de **VM2** où les builds Jenkins seront exécutés)

III) Vérification et Mise en Service de l'Agent

Dans le tableau de bord → Nœuds → VM2-Agent → Journal, on peut voir que l'agent est bien connecté.



IV) Lancer un build grâce à l'agent sur la VM2

Je crée un nouveau job Jenkins pour exécuter des commandes Shell sur VM2 via l'agent configuré

Je coche la case suivante et renseigne le nom de mon agent ou mon label

Au niveau de l'étape de build, j'exécute un script shell à titre d'exemple

Puis je lance le build et observe le résultat sur ma machine VM2 qui contient l'agent Jenkins :