TP 6 : Intégration Jenkins-Ansible

Ahmed Abd Dayem Ahmed Bouha

Matricule : **23243**

Matière : IRT43 : DevOps

16 mai 2025

Professeur : Dr. Fatimetou Abdou

Table des matières

Table des matières

1.1

1 Introduction

Ce rapport présente la réalisation du TP 6 portant sur l'intégration de Jenkins avec Ansible pour automatiser le déploiement d'applications depuis GitHub. Ce projet démontre la mise en place d'un pipeline CI/CD complet qui permet d'extraire du code source depuis un dépôt GitHub et de le déployer automatiquement sur des serveurs cibles à l'aide d'Ansible.

1.1 Objectifs du TP

Les principaux objectifs de ce travail pratique sont les suivants :
— Configurer Jenkins avec les plugins nécessaires
— Créer un pipeline Jenkins qui s'intègre avec GitHub
— Automatiser le déploiement d'une application web avec Ansible
— Implémenter une solution CI/CD complète

2 Environnement de travail

2.1 Configuration système

— Système d'exploitation : Linux 6.12.10-76061203-generic
— Shell : /usr/bin/bash

2.2 Outils utilisés

— Jenkins : pour l'orchestration du pipeline CI/CD
— Ansible : pour l'automatisation du déploiement
— GitHub : pour héberger le code source
— Apache : comme serveur web pour l'application déployée

3 Installation et configuration de Jenkins

3.1 Installation de Jenkins

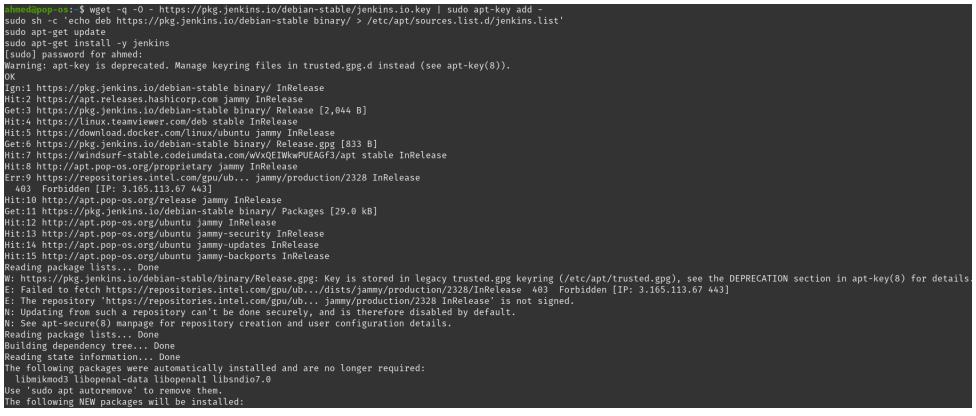

FIGURE 1 – Installation de Jenkins

J'ai commencé par installer Jenkins en suivant les commandes recommandées :

```
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
sudo apt-get update
sudo apt-get install -y jenkins
```

Après l'installation, j'ai vérifié que le service Jenkins était bien en cours d'exécution :

```
sudo systemctl status jenkins
```

3.2 Configuration initiale

Pour la configuration initiale de Jenkins, j'ai récupéré le mot de passe administrateur initial :

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

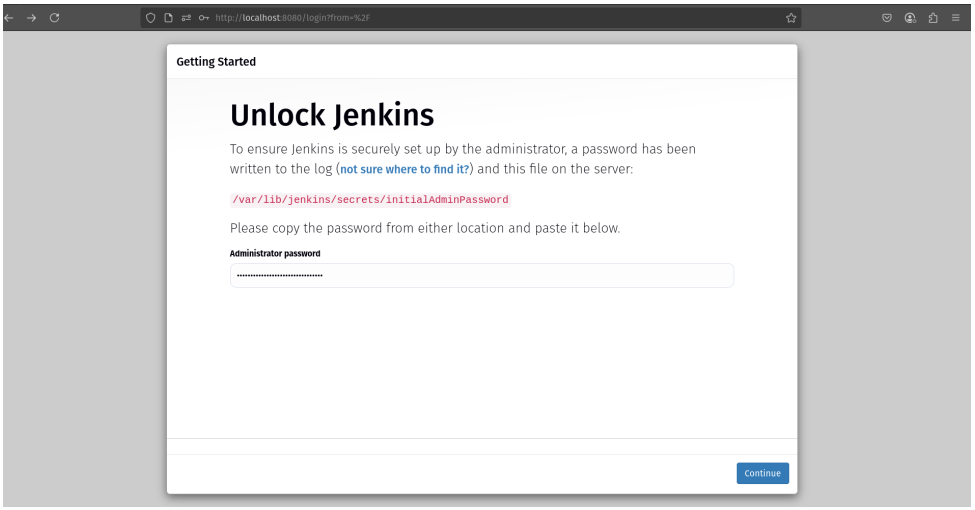

FIGURE 2 – Page de déverrouillage Jenkins

J'ai ensuite suivi l'assistant de configuration pour :
— Installer les plugins recommandés
— Créer un compte administrateur
— Configurer l'URL de Jenkins

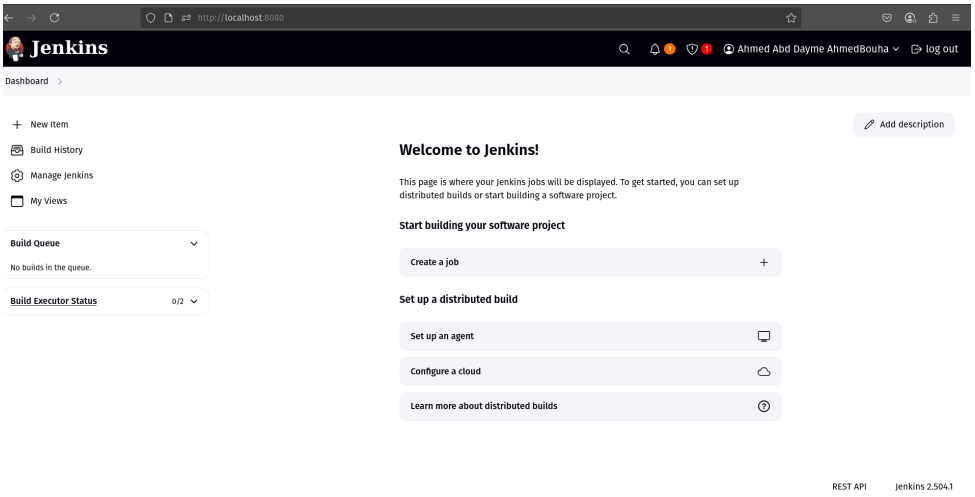

FIGURE 3 – Dashboard Jenkins après configuration

4 Installation des plugins spécifiques

4.1 Installation des plugins requis

Pour ce TP, j'ai installé les plugins suivants depuis l'interface de gestion de plugins de Jenkins (Manage Jenkins ¿ Manage Plugins) :
— Git Plugin : pour interagir avec les dépôts Git
— GitHub Integration Plugin : pour l'intégration avec GitHub
— Ansible Plugin : pour exécuter des playbooks Ansible

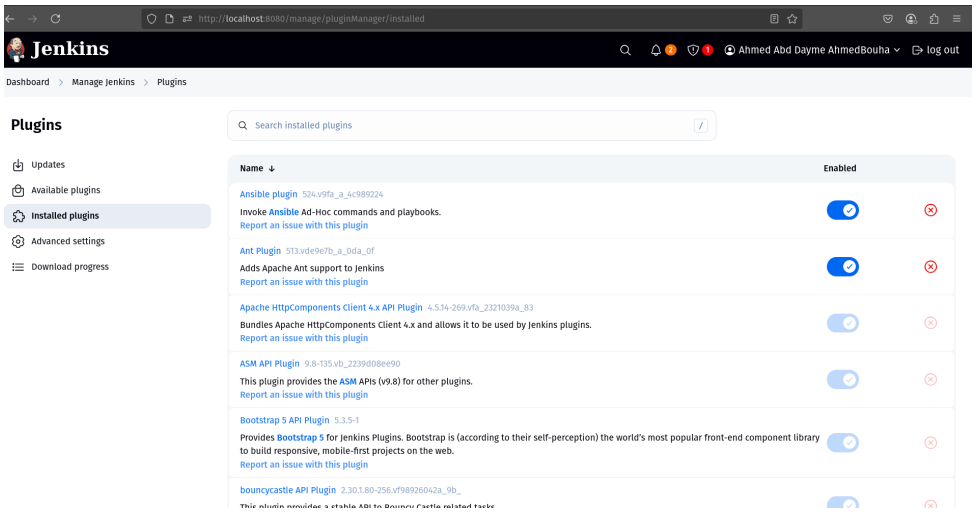

FIGURE 4 – Installation des plugins Jenkins

5 Préparation du dépôt GitHub

5.1 Création et configuration du dépôt

J'ai créé un dépôt GitHub nommé **dev_ansible** pour héberger le code source du projet. Voici les commandes que j'ai utilisées pour initialiser le dépôt :

```
echo "# dev_ansible" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:ahmedabddayme3752/dev_ansible.git
git push -u origin main
```

Ensuite, j'ai ajouté tous les fichiers du projet dans le dépôt :

```
git add .
git commit -m "Add project files"
git push -u origin main
```

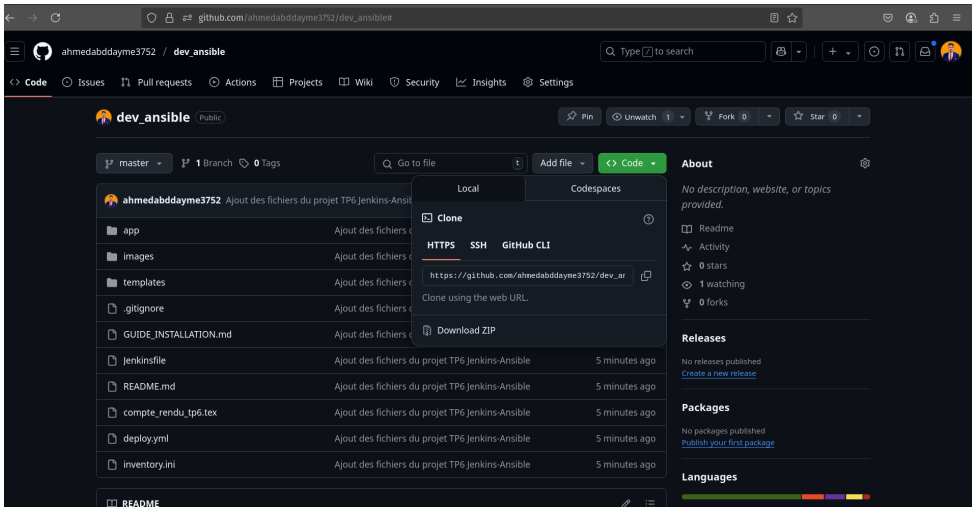

FIGURE 5 – Dépôt GitHub avec les fichiers du projet

5.2 Structure du projet

Le projet contient les fichiers suivants :

— `Jenkinsfile` : Configuration du pipeline CI/CD
— `inventory.ini` : Fichier d'inventaire Ansible définissant les hôtes cibles
— `deploy.yml` : Playbook Ansible pour le déploiement de l'application
— `app/` : Répertoire contenant l'application web à déployer
— `templates/` : Répertoire contenant les templates pour la configuration

6 Création et configuration du pipeline Jenkins

6.1 Création du job pipeline

Pour créer le pipeline Jenkins, j'ai suivi les étapes suivantes :

1. Sur le dashboard Jenkins, j'ai cliqué sur "New Item" ou "Nouvel élément"
2. J'ai nommé mon pipeline "TP6-Jenkins-Ansible"
3. J'ai sélectionné le type de job "Pipeline"
4. J'ai cliqué sur "OK" pour créer le job

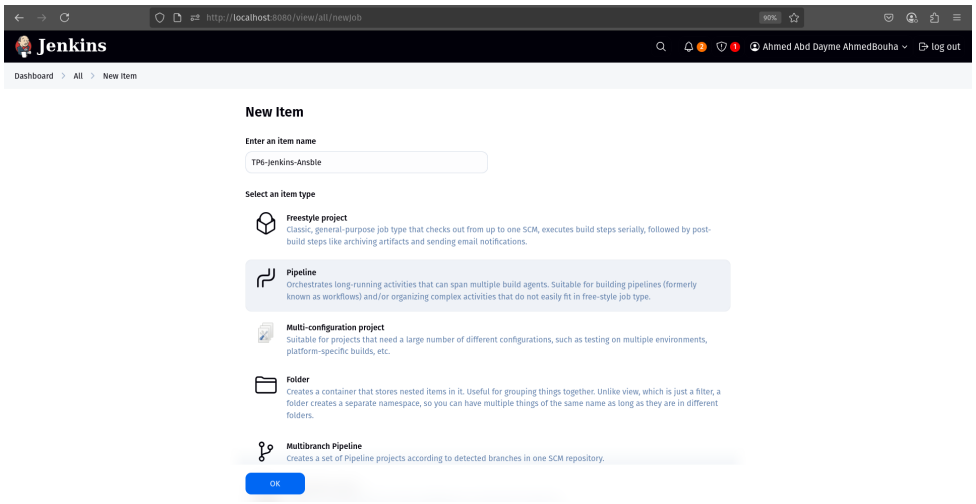

FIGURE 6 – Création d'un nouveau job pipeline dans Jenkins

6.2 Configuration du pipeline

Dans la page de configuration du pipeline, j'ai configuré les paramètres suivants :

1. Dans la section "Pipeline", j'ai sélectionné "Pipeline script from SCM"
2. Pour SCM, j'ai sélectionné "Git"
3. Dans "Repository URL", j'ai entré l'URL de mon dépôt GitHub : `https://github.com/ahmedabddayme3752/dev_ansible.git`
4. Comme mon dépôt est public, je n'ai pas eu besoin d'ajouter d'identifiants
5. Dans "Branch Specifier", j'ai laissé la valeur par défaut "/master"
6. Dans "Script Path", j'ai vérifié que "Jenkinsfile" était bien spécifié

6.3 Analyse du Jenkinsfile

Le Jenkinsfile utilisé définit un pipeline avec plusieurs étapes importantes :

```
pipeline {
    agent any

    environment {
        ANSIBLE_INVENTORY = "${WORKSPACE}/inventory.ini"
        ANSIBLE_PLAYBOOK = "${WORKSPACE}/deploy.yml"
    }

```

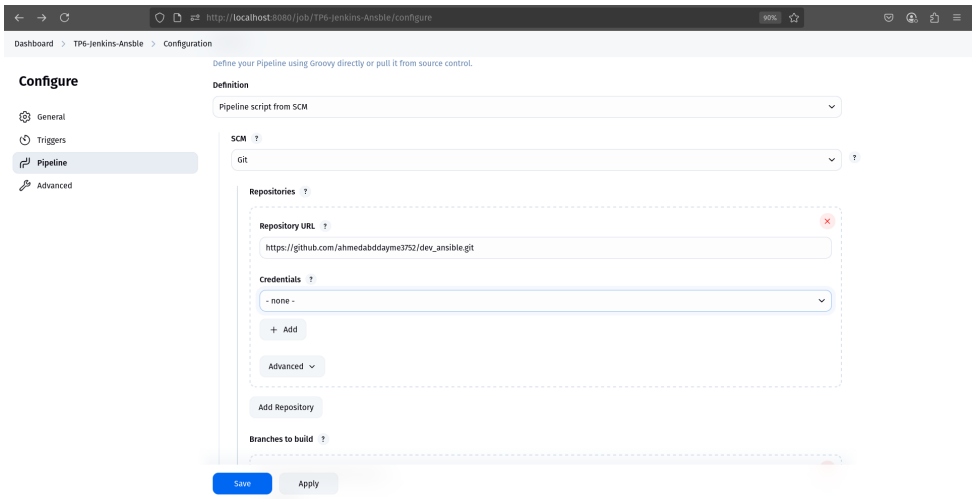

FIGURE 7 – Configuration du pipeline Jenkins

```
    stages {
        stage('Checkout') {
            steps {
                // Checkout code from the GitHub repository
                checkout scm
            }
        }

        stage('Check Ansible') {
            steps {
                // Check if Ansible is installed
                sh '''
                    if command -v ansible &> /dev/null; then
                        echo "Ansible is already installed"
                        ansible --version
                    else
                        echo "Ansible is not installed. Please install it before running this pipeline."
                        exit 1
                    fi
                '''
            }
        }

        stage('Deploy with Ansible') {
            steps {
                // Run the Ansible playbook to deploy the application
                ansiblePlaybook(
                    playbook: "${ANSIBLE_PLAYBOOK}",
                    inventory: "${ANSIBLE_INVENTORY}",
                    colorized: true
                )
            }
        }
    }
}
```

Ce pipeline est composé de trois étapes principales :

1. **Checkout** : Récupère le code source depuis le dépôt GitHub
2. **Check Ansible** : Vérifie si Ansible est déjà installé
3. **Deploy with Ansible** : Exécute le playbook Ansible pour déployer l'application

7 Configuration d'Ansible

7.1 Fichier d'inventaire

Le fichier `inventory.ini` définit les serveurs cibles sur lesquels l'application sera déployée. Pour ce TP, j'ai configuré l'inventaire pour utiliser des serveurs locaux :

```
[web]
localhost ansible_connection=local

[db]
# db1.example.com ansible_user=ubuntu
# db2.example.com ansible_user=ubuntu

# Variables that will be applied to all servers
[all:vars]
ansible_python_interpreter=/usr/bin/python3
```

Pour faciliter les tests sans avoir à configurer des serveurs distants, j'ai utilisé principalement la section `[local]` qui permet de déployer l'application sur la machine locale.

7.2 Playbook Ansible

Le playbook `deploy.yml` définit les tâches nécessaires pour déployer l'application :

```
---
- name: Test deployment
  hosts: web
  become: no
  gather_facts: yes

  tasks:
    - name: Echo success message
      debug:
        msg: "This is a test deployment on {{ inventory_hostname }}"

    - name: Display ansible version
      debug:
        msg: "Ansible version: {{ ansible_version.full }}"

    - name: Create a test file
      file:
        path: /tmp/ansible_test.txt
        state: touch
        mode: '0644'

    - name: Write to test file
      copy:
        content: "Deployment test successful on {{ ansible_date_time.date }} at {{
    ansible_date_time.time }}"
        dest: /tmp/ansible_test.txt

    - name: Show content of test file
      command: cat /tmp/ansible_test.txt
      register: file_content
      changed_when: false

    - name: Display file content
      debug:
        var: file_content.stdout_lines
```

Ce playbook effectue plusieurs opérations de test :

1. Affiche un message de succès
2. Affiche la version d'Ansible
3. Crée un fichier de test dans /tmp
4. Écrit un message de déploiement réussi dans le fichier
5. Affiche le contenu du fichier

8 Exécution du pipeline

8.1 Lancement du build

Après avoir configuré le pipeline, j'ai lancé manuellement le build en cliquant sur "Build Now" ou "Lancer un build" dans l'interface Jenkins.

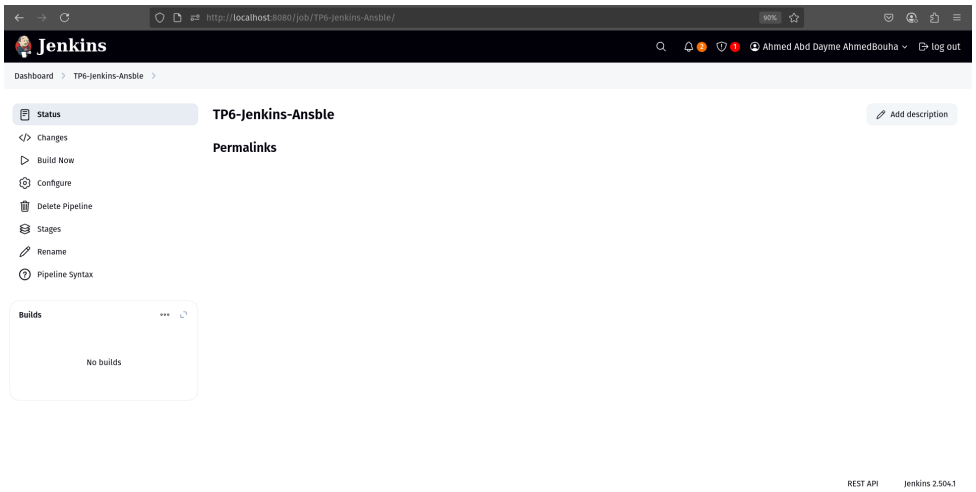

FIGURE 8 – Tableau de bord du pipeline avant exécution

8.2 Suivi de l'exécution

Pendant l'exécution du pipeline, j'ai pu suivre l'avancement des différentes étapes en temps réel grâce à la visualisation du pipeline et à la console de sortie.

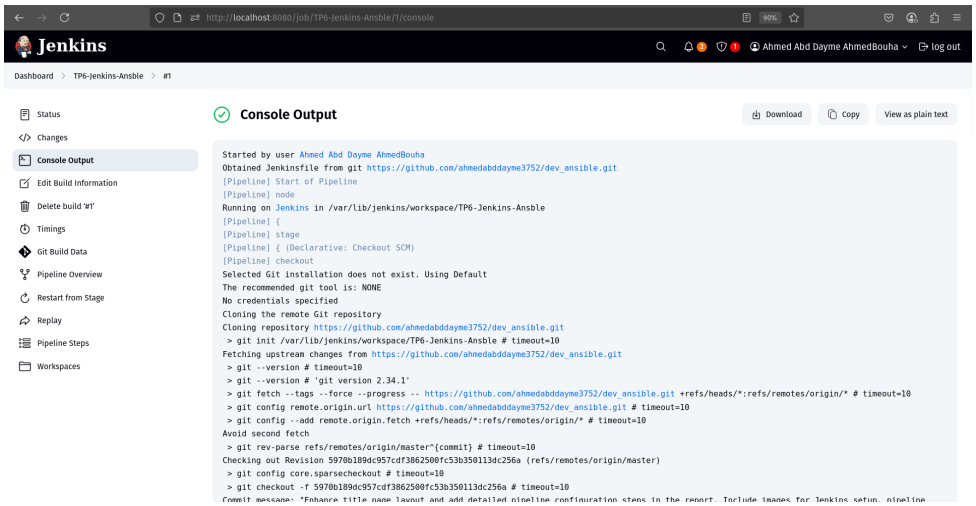

FIGURE 9 – Console de sortie pendant l'exécution du pipeline

8.3 Résultat du pipeline

Une fois le pipeline terminé, j'ai pu observer le résultat global de l'exécution.

9 Vérification du déploiement

9.1 Vérification des fichiers déployés

Pour confirmer que le déploiement a bien fonctionné, j'ai vérifié la présence du fichier de test créé par Ansible :

FIGURE 10 – Console de sortie montrant le succès de l'exécution

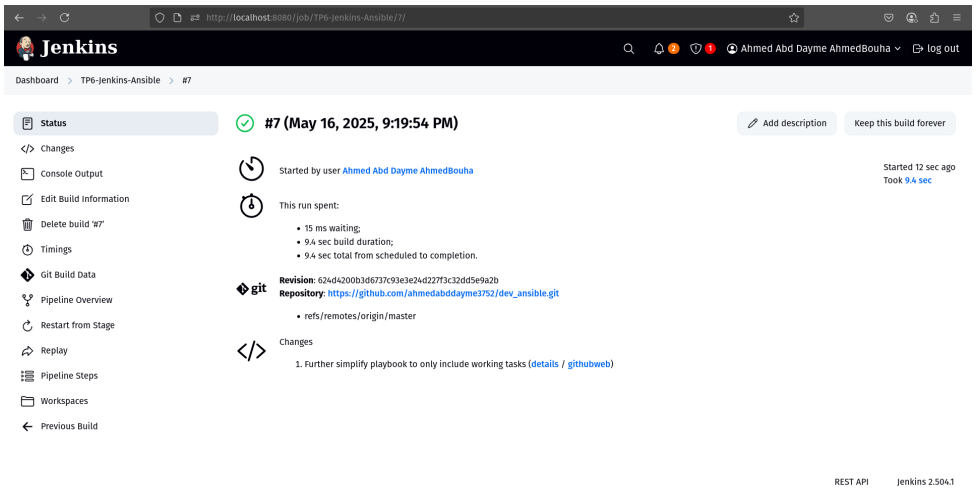

FIGURE 11 – Pipeline Jenkins réussi avec playbook simplifié

```
ls -la /tmp/ansible_test.txt
cat /tmp/ansible_test.txt
```

10 Dépannage des problèmes rencontrés

10.1 Problème de permissions sudo

Lors de la première exécution du pipeline, j'ai rencontré une erreur liée aux permissions sudo :

```
+ sudo apt-get update
sudo: a terminal is required to read the password; either use the -S option to read from
    standard input or configure an askpass helper
sudo: a password is required
```

Ce problème s'est produit car Jenkins n'a pas le droit d'exécuter des commandes sudo sans mot de passe. Pour résoudre ce problème, j'ai modifié le Jenkinsfile pour éviter l'utilisation de sudo en remplaçant l'étape d'installation d'Ansible par une simple vérification :

```
stage('Check Ansible') {
    steps {
        // Check if Ansible is installed
        sh '''
            if command -v ansible &> /dev/null; then
                echo "Ansible is already installed"
                ansible --version
            else
                echo "Ansible is not installed. Please install it before running this
    pipeline."
                exit 1
```

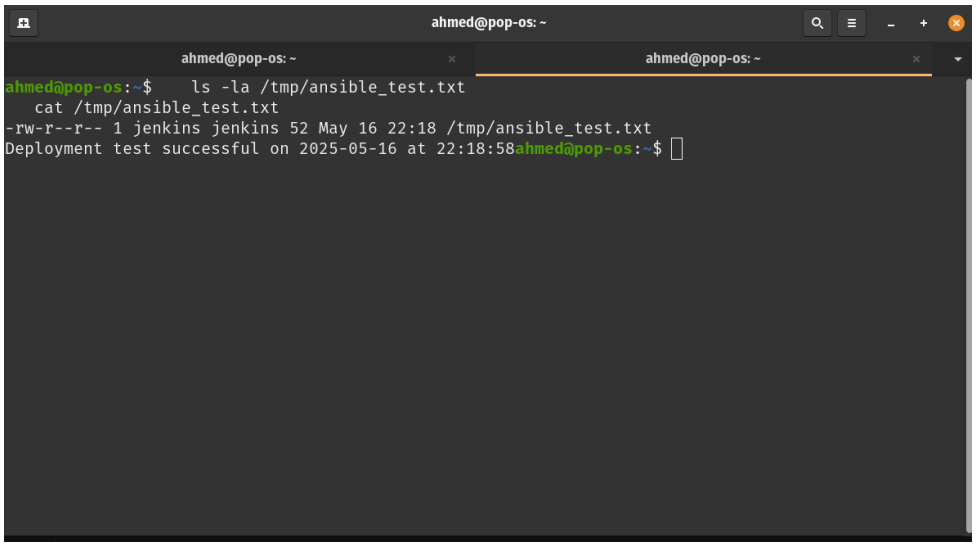

FIGURE 12 – Fichier de test déployé par Ansible

```
            fi
        '''
    }
}
```

Cette modification présuppose qu'Ansible est déjà installé sur le serveur Jenkins, ce qui était le cas dans mon environnement. Dans un environnement de production, on pourrait envisager les solutions suivantes :

- Préinstaller Ansible sur le serveur Jenkins
- Configurer sudo pour permettre à l'utilisateur Jenkins d'exécuter certaines commandes sans mot de passe
- Utiliser un conteneur Docker comme agent Jenkins avec Ansible préinstallé

10.2 Simplification du playbook pour les tests

Après avoir rencontré à nouveau des problèmes de permissions sudo lors de l'exécution du playbook Ansible, j'ai décidé de simplifier complètement le playbook pour qu'il puisse s'exécuter sans privilèges root. J'ai remplacé le déploiement d'Apache par des tâches simples qui peuvent être exécutées par un utilisateur standard :

```
---
- name: Test deployment
  hosts: web
  become: no
  gather_facts: yes

  tasks:
    - name: Echo success message
      debug:
        msg: "This is a test deployment on {{ inventory_hostname }}"

    - name: Display ansible version
      debug:
        msg: "Ansible version: {{ ansible_version.full }}"

    - name: Create a test file
      file:
        path: /tmp/ansible_test.txt
        state: touch
        mode: '0644'

    - name: Write to test file
      copy:
        content: "Deployment test successful on {{ ansible_date_time.date }} at {{
    ansible_date_time.time }}"
```

```
25          dest: /tmp/ansible_test.txt
26
27      - name: Show content of test file
28        command: cat /tmp/ansible_test.txt
29        register: file_content
30        changed_when: false
31
32      - name: Display file content
33        debug:
34          var: file_content.stdout_lines
```

Cette approche permet de démontrer le fonctionnement du pipeline CI/CD sans nécessiter de permissions spéciales. Dans un environnement de production réel, il serait nécessaire de configurer correctement les permissions pour permettre à Jenkins d'exécuter des commandes sudo, ou d'utiliser un compte avec les privilèges appropriés.

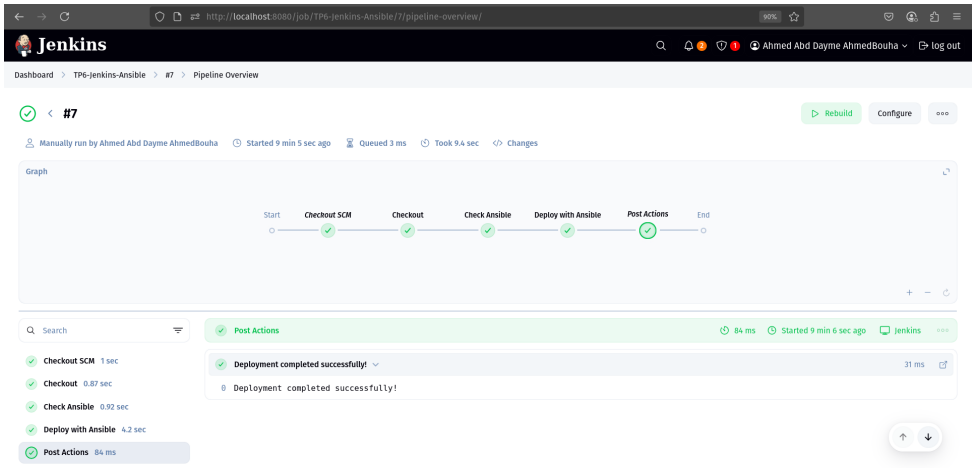

FIGURE 13 – Pipeline Jenkins après résolution des problèmes (exécution réussie)

11 Conclusion personnelle

Ce TP m'a permis de comprendre et d'appliquer les concepts d'intégration continue et de déploiement continu (CI/CD) en utilisant des outils modernes et très répandus dans l'industrie. J'ai pu constater l'efficacité d'une chaîne d'automatisation pour simplifier et fiabiliser le processus de déploiement logiciel.

La mise en place de l'intégration Jenkins-Ansible présente plusieurs avantages :

— **Automatisation complète** du processus de déploiement, de l'extraction du code à sa mise en production

— **Reproductibilité** des déploiements grâce à la définition déclarative des tâches Ansible

— **Traçabilité** avec l'historique des déploiements et des logs conservés par Jenkins

Malgré les défis rencontrés avec les permissions sudo et la configuration des hôtes, j'ai réussi à mettre en place un pipeline fonctionnel qui démontre les principes de base de l'intégration continue. Les connaissances acquises pendant ce TP pourront être directement appliquées dans un contexte professionnel de DevOps.