Demo Project: Ansible Integration in Jenkins

Project Description

In this project, we will:

- Create and configure a **dedicated server** for Jenkins.
- Create and configure a **dedicated server** for the Ansible Control Node.
- Write an **Ansible Playbook** to configure two EC2 instances.
- Add **SSH key credentials** in Jenkins for both the Ansible Control Node and the Ansible Managed Nodes.
- Configure **Jenkins to execute the Ansible Playbook** on the Ansible Control Node as part of a CI/CD pipeline.

The **Jenkins pipeline** will:

1. Connect to the remote **Ansible Control Node**.
2. Copy the **Ansible playbook** and configuration files to the Ansible Control Node.
3. Copy the **SSH keys** for the Ansible Managed Nodes to the Ansible Control Node.
4. Install **Ansible, Python3, and Boto3** on the Ansible Control Node.
5. Execute the **Ansible playbook remotely** to configure the two EC2 instances.

Step 1: Prepare Ansible Server (Control Node)

1. **Go to DigitalOcean** and create a new droplet.
2. **Choose Ubuntu** as the operating system.
3. **Select Basic Shared CPU**:
    - **Regular Disk Type: SSD**
    - **2 GB RAM / 2 CPU**
4. **Authentication Method:** Use an **existing SSH key**.
5. Click **Create Droplet** and **rename it to** `ansible-server`.
6. Copy the **Public IP Address**.
7. **SSH into the Ansible server:** `ssh root@<public IP>`
8. Install Anisble:

    ```
    apt update
    apt install ansible-core -y
    ```

    - Verify installation: `ansible`
9. **Install Python and Boto3:** `apt install python3-boto3 -y`
10. **Add AWS credentials for Ansible to connect to EC2 instances:**
    - On your local machine: `cat .aws/credentials`
    - Copy the **AWS access keys** and paste them into the **Ansible server**:
        - `mkdir .aws`
        - `cd .aws`
        - `vim credentials`

Paste the following:

```
[default]
aws_access_key_id = XXXXXXXXXXXXXXXXXXXXXX
aws_secret_access_key = XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

11. Save and exit

Step 2: Create 2 EC2 Instances (Managed Server) to be managed by Ansible

1. **Go to AWS Console → EC2**.
2. **Launch 2 EC2 instances** with default settings.
3. **Create a new key pair:**
   - **Name:** `ansible-jenkins`
   - Download the key locally (`ansible-jenkins.pem`).
4. **Wait until the instances are in Running state**.

Step 3: Copy Files from Jenkins to Ansible Server

1. **Clone the Git repository:**

   `git clone` `git@gitlab.com` `:twn-devops-projects/ansible/java-maven-app.git`
2. From the `main` branch create a new branch: `git checkout -b feature/ansible`
3. **Copy Ansible configuration files:**
   - Create a directory named `ansible`.
   - Copy **inventory_aws_ec2.yaml** and **ansible.cfg** from an existing Ansible project.

4. **Ensure `ansible.cfg` is configured properly:**

```
[defaults]
host_key_checking = False
inventory = hosts
# inventory = inventory_aws_ec2.yaml

interpreter_python = /usr/bin/python3.9

enable_plugins = aws_ec2

remote_user = ec2-user
private_key_file = /home/eb/.ssh/id_rsa
```

Step 4: Create Ansible Playbook

- **Create `my-playbook.yaml` with the following content:**

```
- name: Install Docker
  hosts: all
  become: yes
  tasks:
    - name: Install Docker
      yum:
        name: docker
        update_cache: yes
        state: present
    - name: Start docker daemon
      systemd:
        name: docker
        state: started

- name: Install Docker-compose
  hosts: all
```

```
  tasks:
    - name: Create docker-compose directory
      file:
        path: ~/.docker/cli-plugins
        state: directory
    - name: Get architecture of remote machine
      shell: uname -m
      register: remote_arch
    - name: Install docker-compose
      get_url:
        url: "https://github.com/docker/compose/releases/latest/download/dock
er-compose-linux-{{ remote_arch.stdout }}"
        dest: ~/.docker/cli-plugins/docker-compose
        mode: +x
```

Step 5: Configure SSH Keys in Jenkins

1. **Go to Jenkins Dashboard → Manage Jenkins → Credentials → System → Global credentials → Add Credentials**.
2. **Create an SSH key for the Ansible server:**
   - **Kind:** SSH Username with private key
   - **ID:** `ansible-server-key`
   - **Username:** `root`
   - **Private Key:** Paste from `cat ~/.ssh/id_rsa`

   Note: if the key starts with `-----BEGIN OPENSSH PRIVATE KEY-----` and then ends with `-----END OPENSSH PRIVATE KEY-----` it is because its with the new ssh version format. Conversion is needed to convert to the classic SSH format, so Ansible doesn't return an error.

   To convert:

   1. `cp ~/.sshid_rsa ~/.sshssh_key_rsa_format`
   2. `ssh-keygen -p -f .ssh/id_rsa -m pem -P "" -N ""`

3. Copy and then paste within Private Key in Jenkins create credentials dashboard.

3. **Create an SSH key for the EC2 instances:**
   - **Kind:** SSH Username with private key
   - **ID:** `ec2-server-key`
   - **Username:** `ec2-user`
   - **Private Key:** Copy from `cat ~/Downloads/ansible-jenkins.pem`

The difference between these two keys is:

ansible-server-key	root	SSH Username with private key
ec2-server-key	ec2-user	SSH Username with private key

ansible-server-key: to connect to the ansible server from Jenkins

ec2-server-key: copy contents to the ansible server

Step 6: Create Jenkins Pipeline

1. **Go to Jenkins Dashboard → New Item**.
2. **Enter Pipeline Name:** `ansible-pipeline`.
3. **Choose Type:** Pipeline.
4. **Scroll down to Pipeline → Definition → Pipeline script from SCM**.
5. **Set SCM:** Git.
6. **Set Repository URL:** `https://gitlab.com/twn-devops-projects/ansible/java-maven-app`.
7. **Set Branch Specifier:** `feature/ansible`.
8. **Click Save.**

Step 7: Configure Jenkinsfile

1. In Jenkins → Manage Jenkins → Plugins → Install “SSH Pipeline steps” which provides SSH facilities such as command execution or file transfer for continuous delivery
2. Create a `Jenkinsfile` with the following content:

```
pipeline {
  agent any
  environment {
    ANSIBLE_SERVER = "147.182.128.129"
  }
  stages {
    stage("copy files to ansible server") {
      steps {
        script {
          echo "copying all neccessary files to ansible control node"
          sshagent(['ansible-server-key']) {
            sh "scp -o StrictHostKeyChecking=no ansible/* root@${ANSIBLE_SER
VER}:/root" // Copy everything from the ansible folder from this project into th
e ansible server /root location

            withCredentials([sshUserPrivateKey(credentialsId: 'ec2-server-key', ke
yFileVariable: 'keyfile', usernameVariable: 'user')]) {
              sh 'scp $keyfile root@$ANSIBLE_SERVER:/root/ssh-key.pem'
            }
          }
        }
      }
    }
    stage ("execute ansible playbook") {
      steps {
        script {
          echo "calling ansible playbook to configure ec2 instances"
          def remote = [:]
```

```
        remote.name = "ansible-server"
        remote.host = ANSIBLE_SERVER
        remote.allowAnyHosts = true

        withCredentials([sshUserPrivateKey(credentialsId: 'ansible-server-key',
keyFileVariable: 'keyfile', usernameVariable: 'user')]) {
          remote.user = user
          remote.identityFile = keyfile
          // sshScript remote: remote, script: "prepare-ansible-server.sh"
          sshCommand remote: remote, command: "ansible-playbook my-playb
ook.yaml"
        }
      }
    }
  }
}
```

Optional create `prepare-ansible-server.sh`:

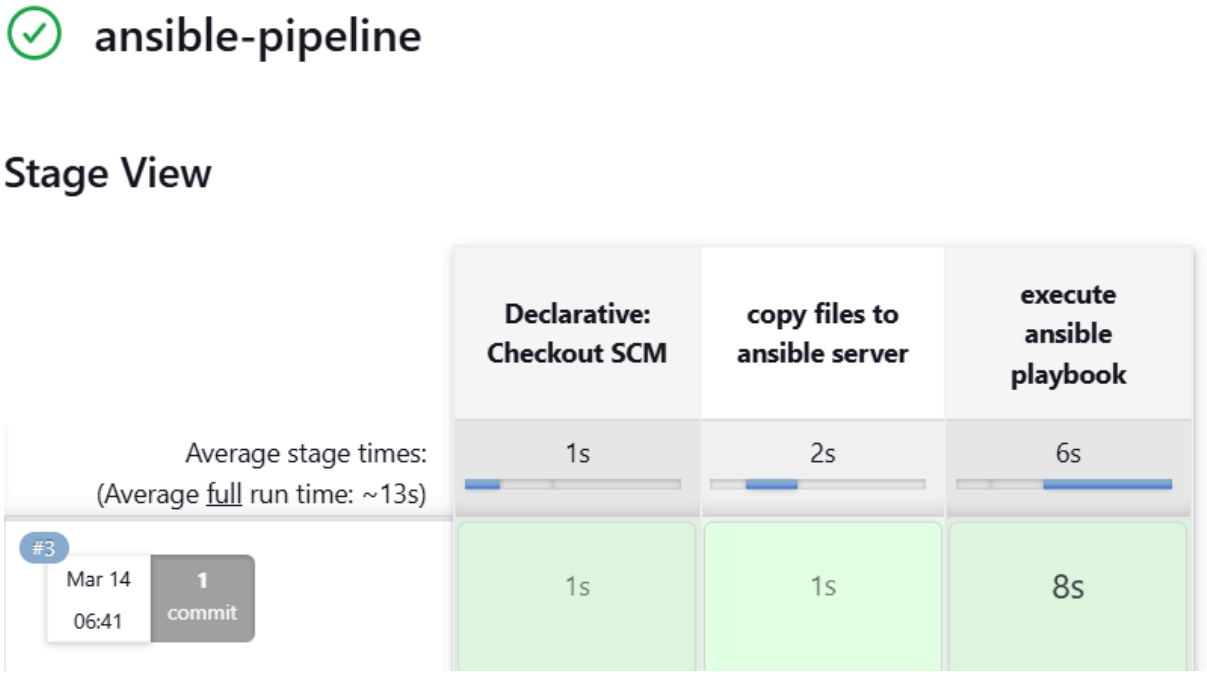
```
#!/usr/bin/env bash

apt update
apt install ansible -y
apt install python3-boto3
```

- Commit and Push to the repo

Step 8: Run the Pipeline

1. **Go to Jenkins Dashboard → Click** `Build Now`.
2. **Monitor the pipeline execution**.



Step 9: Clean up

Remember to destroy your Ansible and Jenkins servers at the end