Using YAML Files to Define and Manage Configurations in Ansible Playbooks

Table of Contents

Description

This document explains how to use **YAML files** to define configurations in **Ansible playbooks**. YAML (Yet Another Markup Language) is a human-readable data format that simplifies writing and managing configurations in Ansible, making automation easier and more structured.

Problem Statement

Managing system configurations and deployment tasks across multiple environments (development, staging, production) can be time-consuming. **Ansible playbooks** in **YAML format** offer a clean, concise way to automate and manage these tasks across multiple systems.

Prerequisites

Completion of all previous lab guides (up to Lab Guide-07) is required before proceeding with Lab Guide-08.

Software Required

- **Ansible**: Installed on your control machine (Linux/macOS or Windows Subsystem for Linux).
- **SSH**: Set up to connect from the control machine to target nodes.
- **Python**: Installed on both the control machine and target nodes.
- **TodoAPP_MYSQl**: To download the source folder **click here**

Hardware Requirement

- Minimum of 2 GB RAM on the control machine.
- SSH access to target systems.

Implementation Steps

Step-1: Define a Basic Ansible Playbook in YAML

An Ansible playbook defines tasks and configurations to be executed on target systems. Let's create a **simple playbook** in YAML to automate the deployment of **TodoApp**.

`todoapp_playbook.yml` - Basic Ansible Playbook

```yaml
---
- name: Deploy TodoApp
  hosts: all
  become: yes

  tasks:
    - name: Install Docker
      apt:
        name: docker.io
        state: present
        update_cache: yes

    - name: Start Docker service
      service:
        name: docker
        state: started
        enabled: yes

    - name: Pull TodoApp Docker image
      docker_image:
        name: my_todoapp
        source: pull

    - name: Run TodoApp container
      docker_container:
        name: todoapp
        image: my_todoapp
        state: started
        ports:
          - "8081:8081"

    - name: Ensure container is running
      docker_container_info:
        name: todoapp
      register: todoapp_status

    - name: Debug container status
      debug:
        var: todoapp_status
```

Explanation:

This YAML file is an **Ansible playbook** that deploys a Todo application using Docker. Here's a breakdown of each section:

1. **Playbook Metadata**:
    - **name**: `Deploy TodoApp` – This playbook deploys the TodoApp container.
    - **hosts**: `all` – The playbook runs on all hosts in the Ansible inventory.
    - **become**: `yes` – Enables privilege escalation to `sudo` for tasks requiring root access.
2. **Tasks**:
    - **Install Docker**:
        - Uses the `apt` module to install Docker (`docker.io`) on the host, ensuring it's present and the package cache is updated.
    - **Start Docker service**:
        - Uses the `service` module to start Docker if it's not already running and enables it to start on boot.
    - **Pull TodoApp Docker image**:
        - Uses the `docker_image` module to pull the Docker image named `my_todoapp` from Docker Hub or a configured registry.
    - **Run TodoApp container**:
        - Uses the `docker_container` module to run a container named `todoapp` using the `my_todoapp` image.
        - Maps port `8081` on the host to port `8081` in the container to make the application accessible.
    - **Ensure container is running**:
        - Uses `docker_container_info` to check the status of the `todoapp` container, storing the result in the `todoapp_status` variable.
    - **Debug container status**:
        - Uses `debug` to print the `todoapp_status` information, showing details about the container (useful for verification).

Step-2: Run the Playbook to Automate Configurations

Once the playbook is created, run it using the following command:

```
ansible-playbook -i inventory todoapp_playbook.yml
```

- **inventory**: This file contains the list of target systems (hosts) where the playbook will run.

Sample `inventory` file:

```
[servers]
192.168.1.100 ansible_user=ubuntu ansible_ssh_private_key_file=~/.ssh/id_rsa
192.168.1.101 ansible_user=ubuntu ansible_ssh_private_key_file=~/.ssh/id_rsa
```

This inventory defines the servers (or nodes) Ansible will connect to and execute the playbook on.

References

- Ansible Documentation: https://docs.ansible.com/
- YAML Syntax Guide: https://yaml.org/