**Ansible Exercises Handbook (Basic to Intermediate)**

**Chapter 1: Introduction to Ansible**

**Exercise 1: Install Ansible**

* **Objective**: Install Ansible on a local machine.
* **Steps**:
  1. Install Ansible on your local machine using your OS's package manager or by following official documentation.
  2. Verify installation by running the command ansible --version.

**Exercise 2: Setup a Simple Inventory File**

* **Objective**: Create a basic Ansible inventory file.
* **Steps**:
  1. Create a text file inventory.ini and define two groups of hosts (webservers, dbservers).
  2. Add IP addresses of your test servers under each group.
  3. Use the ansible all -m ping -i inventory.ini command to check connectivity.

**Chapter 2: Basic Ansible Commands and Playbooks**

**Exercise 3: Ping All Hosts**

* **Objective**: Use Ansible to ping all hosts in the inventory file.
* **Steps**:
  1. Use the command ansible all -m ping -i inventory.ini to verify connectivity to all hosts.
  2. Troubleshoot any issues, if the ping fails.

**Exercise 4: Create a Simple Playbook**

* **Objective**: Write a basic Ansible playbook to install Apache on a web server.
* **Steps**:
  1. Create a playbook install\_apache.yml that includes:

---

- hosts: webservers

become: true

tasks:

- name: Install Apache

apt:

name: apache2

state: present

* 1. Run the playbook with ansible-playbook install\_apache.yml -i inventory.ini.

**Chapter 3: Managing Packages and Services**

**Exercise 5: Install Multiple Packages**

* **Objective**: Install multiple packages on all hosts in an inventory.
* **Steps**:
  1. Update the install\_apache.yml playbook to install curl and git in addition to apache2.
  2. Run the playbook and verify that all packages are installed.

**Exercise 6: Managing Services**

* **Objective**: Use Ansible to start, stop, and enable a service.
* **Steps**:
  1. Create a playbook manage\_services.yml to ensure that Apache is started and enabled.

---

- hosts: webservers

become: true

tasks:

- name: Start and enable Apache

service:

name: apache2

state: started

enabled: true

* 1. Run the playbook and check the status of the Apache service.

**Chapter 4: Variables, Templates, and Loops**

**Exercise 7: Using Variables in Playbooks**

* **Objective**: Implement variables in Ansible playbooks.
* **Steps**:
  1. Create a playbook install\_apache\_with\_version.yml that uses a variable to specify the Apache version.
  2. Define the variable apache\_version in the playbook and use it in the installation task.

---

- hosts: webservers

become: true

vars:

apache\_version: "2.4"

tasks:

- name: Install specific version of Apache

apt:

name: "apache2={{ apache\_version }}"

state: present

* 1. Run the playbook and check if the correct version is installed.

**Exercise 8: Use Jinja2 Templates**

* **Objective**: Use Ansible's template module to deploy a configuration file.
* **Steps**:
  1. Create a template.j2 file with Apache’s configuration settings:

Listen {{ apache\_port }}

DocumentRoot {{ apache\_docroot }}

* 1. In your playbook, define the variables apache\_port and apache\_docroot, and use the template module to copy the template to your server.

yaml

Copy

---

- hosts: webservers

become: true

vars:

apache\_port: 8080

apache\_docroot: "/var/www/html"

tasks:

- name: Deploy Apache configuration

template:

src: template.j2

dest: /etc/apache2/sites-available/000-default.conf

* 1. Run the playbook and verify the Apache configuration.

**Exercise 9: Looping Through Items**

* **Objective**: Create a loop to install multiple packages.
* **Steps**:
  1. Modify the playbook install\_apache.yml to install multiple packages (e.g., curl, git, vim) using a loop.

---

- hosts: all

become: true

vars:

packages:

- curl

- git

- vim

tasks:

- name: Install packages

apt:

name: "{{ item }}"

state: present

loop: "{{ packages }}"

* 1. Run the playbook and ensure that all the packages are installed.

**Chapter 5: Advanced Topics and Best Practices**

**Exercise 10: Roles**

* **Objective**: Create a role for Apache installation.
* **Steps**:
  1. Create a role apache with tasks for installation, configuration, and starting the service.
  2. Structure your project directory:

css

Copy

├── playbooks/

│ └── apache\_install.yml

├── roles/

│ └── apache/

│ ├── tasks/

│ │ └── main.yml

│ ├── templates/

│ └── vars/

└── inventory.ini

* 1. Write the tasks/main.yml to install Apache, configure it, and start the service.
  2. Use the role in your playbook:

---

- hosts: webservers

become: true

roles:

- apache

**Exercise 11: Use Handlers**

* **Objective**: Use handlers to restart services only when necessary.
* **Steps**:
  1. Modify the playbook apache\_install.yml to use a handler for restarting Apache only when the configuration changes.

---

- hosts: webservers

become: true

tasks:

- name: Install Apache

apt:

name: apache2

state: present

notify:

- restart apache

handlers:

- name: restart apache

service:

name: apache2

state: restarted

**Chapter 6: Debugging and Troubleshooting**

**Exercise 12: Debugging Playbooks**

* **Objective**: Learn to debug Ansible playbooks using the debug module.
* **Steps**:
  1. Add a debug task to your playbook install\_apache.yml to print the value of apache\_version.

- name: Debug apache version

debug:

var: apache\_version

* 1. Run the playbook and observe the output.

**Exercise 13: Error Handling**

* **Objective**: Implement error handling in playbooks.
* **Steps**:
  1. Add a block and rescue in your playbook to handle errors.

---

- hosts: webservers

become: true

tasks:

- block:

- name: Install Apache

apt:

name: apache2

state: present

rescue:

- name: Handle installation failure

debug:

msg: "Apache installation failed!"