Ansible Automation: Playbooks, Variables, and Debugging"

1. Ansible:

- Automation Tool:
  - Ansible is an open-source automation tool used for configuration management, application deployment, task automation, and orchestration.
- Declarative Approach:
  - Ansible follows a declarative approach, where the user specifies the desired state of the system, and Ansible ensures that the system reaches and maintains that state.

2. Ansible Playbook: `main.yml`

- Playbook Structure:
  - A playbook is a YAML file that defines a set of tasks to be executed on hosts.
- Hosts:
  - The `hosts: all` line specifies that the playbook targets all hosts.
- Variables:
  - The `vars_files` section includes an external variable file (`var.yml`) for storing variables used in the playbook.
- Tasks:
  - The playbook consists of tasks, each with a specific purpose.
- Package Module:
  - The `package` module is used to manage packages. In this case, it installs a specified web package.
- Debug Module:
  - The `debug` module is used to print information during playbook execution. It is used here to display the 'failed' status.
- Copy Module:

        - The `copy` module copies content to a specified destination. Here, it creates an `index.html` file.
- Service Module:
    - The `service` module manages system services. It starts or stops the specified web service based on the 'isStart' variable.

3. External Variable File: `var.yml`

- Variable Definition:
    - The `var.yml` file defines variables (`webPackage`, `docRoot`, `webService`, `isStart`) used in the main playbook.
- Variable Values:
    - Variables are assigned specific values (`httpd`, `/var/www/html`, `httpd`, `started`) to be used in the playbook.

4. Debug Playbook: `debug.yml`

- Debugging with Ansible:
    - The `debug.yml` playbook is used for debugging purposes on the localhost.
- Debug Module (Task 1):
    - The first task uses the `debug` module to display the value of the variable `x`.
- Debug Module (Task 2):
    - The second task demonstrates string interpolation in Ansible, displaying a custom message with interpolated variables.

5. Commands for Execution:

- Syntax Check:
    - `ansible-playbook --syntax-check main.yml` checks the syntax of the main playbook.
- Run Playbook with Extra Variables:
    - `ansible-playbook --extra-vars="isStart=stopped" main.yml` runs the main playbook with an extra variable (`isStart`) specified during execution.

Debug Playbook: `debug.yml`

```
# debug.yml
```

```
# Playbook for debugging purposes targeting localhost
- hosts: localhost

  # Define variables for debugging
  vars:
    x: 5
    myname: "ayush"

  # Task 1: Debug the variable 'x'
  tasks:
    - debug:
        var: x

- hosts: localhost

  # Define variables for debugging
  vars:
    x: 5
    myname: "ayush"
```

```
  # Task 2: Debug a custom message with interpolated variables

  tasks:

    - debug:

        msg: "my name is {{ myname }} and the value of x is {{ x }}"
```

Ansible Playbook: `main.yml`

```
# main.yml

# Playbook targets all hosts
- hosts: all

  # Include external variable file
  vars_files:
    - "var.yml"

  # Tasks to be executed
  tasks:
    # Task 1: Install the specified web package
    - name: Install web package
      package:
        name: "{{ webPackage }}"
        state: present
      register: packageRegister
      ignore_errors: true

    # Task 2: Debug the 'failed' status of the package installation
    - name: Debug failed status
      debug:
        var: packageRegister['failed']

    # Task 3: Install httpd package if the web package installation fails
    - name: Install httpd if web package installation fails
      package:
        name: "httpd"
        state: present
```

```
    when: packageRegister['failed']

  # Task 4: Copy content to index.html
  - name: Copy content to index.html
    copy:
      content: "i am ayush"
      dest: "{{ docRoot }}/index.html"

  # Task 5: Manage the web service based on 'isStart' variable
  - name: Manage web service
    service:
      name: "{{ webService }}"
      state: "{{ isStart }}"
```

External Variable File: `var.yml`

```
# var.yml

# Define variables used in the main playbook
webPackage: "httpd"
docRoot: "/var/www/html"
webService: "httpd"
isStart: started

# Check syntax of the main playbook
ansible-playbook --syntax-check main.ym
# Run the main playbook
ansible-playbook  main.yml

# Run the main playbook with extra variables
ansible-playbook --extra-vars="isStart=stopped" main.yml
```