Ansible Playbook - YAML file

This document provides a comprehensive guide to understanding my Ansible playbook, which has been configured to run solely on my local host. As a beginner in Ansible, I aim to explain each input and output line in detail, shedding light on how the playbook operates and interacts with the local environment. This guide is designed to help new users grasp the basics of Ansible automation and understand how to effectively use it for local tasks.

1.Playbook to simply print a message

```
---
- hosts: localhost
 tasks:
  - name: print message
    debug:
     msg: HEllo there
```

Here's an explanation of each step in Ansible playbook:

1. " — "

This is the start of the YAML file, which is the format Ansible playbooks are written in. It signifies the beginning of the playbook.

2. " - hosts: localhost "

This line specifies the target hosts where the tasks in the playbook will be executed. In this case, `localhost` is used, which means the playbook will run on our local machine.

3. " tasks: "

This section defines a list of tasks that will be performed on the target host(s). Each task has a specific purpose, such as running commands, copying files, or managing configurations.

4. " - name: print message "

This is the beginning of a task. Each task in Ansible can have a `name` attribute, which describes what the task does. Here, the task is named "print message," which gives a brief description of the task's purpose.

name : A human-readable description of the task to make the playbook more understandable. It doesn't affect functionality but helps in reading the output.

5. " debug: "

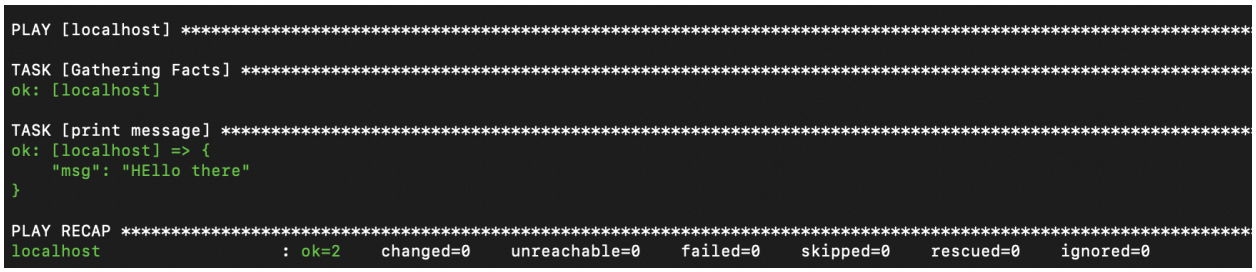
This is the Ansible module being used in this task. The `debug` module is a built-in Ansible tool used to display messages in the playbook's output. It's useful for troubleshooting and printing variable values or messages during execution.

6. “ msg: HEllo there ”

This is the argument passed to the `debug` module. It tells the module what message to print. In this case, the message "HEllo there" will be displayed in the output when the playbook is run.”msg” is the parameter for debug module.

Command to run the ansible playbook is : `ansible-playbook filename.yml`

Here is the output:



4. PLAY [localhost]

This indicates that Ansible is starting the play on the `localhost` host.

- PLAY : This signifies the start of the play.
- [localhost] : The play is targeting `localhost`, which is the only host available.

5. TASK [Gathering Facts]

This task automatically gathers facts about the target host (`localhost`). Facts are system-related variables such as OS type, memory, CPU, and more, which Ansible collects before executing the tasks.

- **ok: [localhost]:** This message means that the task of gathering facts was successful, and the `localhost` host is functioning properly.

6. TASK [print message]

This is the task that was defined in your playbook. It’s using the `debug` module to print the message "HEllo there".

- **`ok: [localhost] => {"msg": "HEllo there"}`**: This output shows that the message task was successfully executed, and the message `"HEllo there"` was displayed for the `localhost`.

7. PLAY RECAP

At the end of the play, Ansible provides a recap of what happened.

- `localhost`: The name of the host where the playbook was executed (in this case, `localhost`).

The recap includes the following:

- ok=2: Two tasks were successfully executed (gathering facts and printing the message).
- changed=0: No changes were made to the system, as the tasks were informational.

- unreachable=0: No hosts were unreachable, meaning Ansible was able to communicate with all intended hosts (just `localhost` here).
- failed=0: No tasks failed during execution.
- skipped=0: No tasks were skipped.
- rescued=0: No tasks required rescuing (error recovery).
- ignored=0: No tasks were ignored.

2. Playbook to install homebrew packages for macOS

```
---
- name: Install and manage Homebrew package on macOS
 hosts: localhost
 tasks:
   - name: Install a list of brew packages
     homebrew:
      name: "{{ item }}"
      state: present
     with_items:
      - git
      - wget
      - python
      - node
```

Here's an explanation of each step in your updated playbook:

1. `---`

This marks the start of the YAML file, which is the format used for Ansible playbooks.

2. `- name: Install and manage Homebrew package on macOS`

This defines the playbook's name, which describes the general purpose of the playbook. In this case, the playbook is designed to install and manage Homebrew packages on a macOS machine.

3. `hosts: localhost`

This specifies the target hosts for the playbook. Since it's set to `localhost`, the tasks will be executed on our local machine.

4. `tasks:`

This begins the section where the list of tasks is defined. These tasks will be performed on the target hosts (in this case, `localhost`).

5. `- name: Install a list of brew packages`

This is a specific task within the playbook. The task will install several packages using Homebrew, a package manager for macOS.

- `**name**`: A descriptive title for the task that indicates it will install multiple Homebrew packages.

6. `**homebrew**:`

This is the Ansible module being used to manage packages with Homebrew. The `homebrew` module allows you to install, remove, or manage packages on macOS via the Homebrew package manager.

7. `name: "{{ item }}"`

This defines a dynamic value using a variable (`{{ item }}`). It allows you to pass each item in a list (defined below) to the `homebrew` module one by one.

- `{{ **item** }}`: A placeholder for each package in the list. Ansible will loop through the list of packages and apply them to the `homebrew` module.

8. `state: present`

This ensures that the packages listed are installed on the system. If they are not present, Ansible will install them.

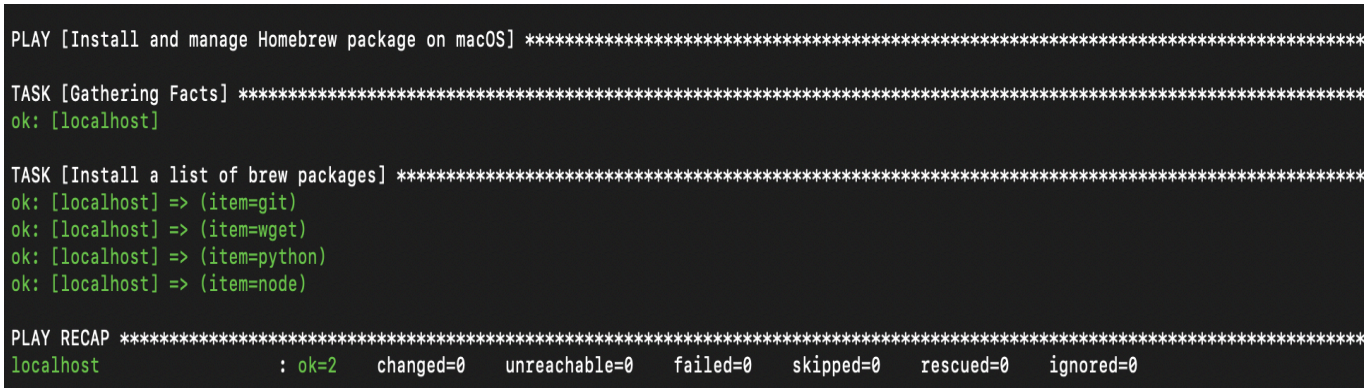
9. `with_items:`

This indicates that a list of items (in this case, packages) will be iterated over. Ansible will loop through each item in the list and apply the `homebrew` task for each one.

10. List of Packages:

- git: A version control system to track changes in source code.
- wget: A command-line utility for downloading files from the web.
- python: The Python programming language.
- node: Node.js, a JavaScript runtime used for server-side scripting.

Here is the output



TASK [Install a list of brew packages]:

This task installs the list of Homebrew packages (`git`, `wget`, `python`, and `node`) on your local machine.

ok: [localhost] => (item=git): The package `git` was checked and found to be installed, so no changes were made.
-ok: [localhost] => (item=wget): The package `wget` was already installed, so no changes were made.
-ok: [localhost] => (item=python): The package `python` was found and required no action.
-ok: [localhost] => (item=node): The package `node` was already present and did not need to be installed.
s
- The playbook successfully completed.
- Ansible checked for the presence of the Homebrew packages (`git`, `wget`, `python`, and `node`) and found that they were already installed on your machine, so no changes were made.