joindevops.com

Ansible Interview Questionnaire

Part -1

> **Answers are not bookish definitions. Hope it is useful for you. Try to give detailed answers in interviews, no statements or single word answers.**

Q: What is configuration management?

A: Configuration management is a way of managing servers for consistency, scalability and reliability. This is a master node architecture. I will take the example of Ansible for this to explain.

We will have an Ansible server where Ansible is installed. We can call servers as nodes. Ansible connects to all the servers using SSH protocol. We can write playbooks that are version controlled in GIT. Ansible can run these playbooks against the nodes. Ansible can connect to multiple servers at a time to achieve scalability. Since it is configuration as a code, we can run the same code in multiple environments to achieve consistency.
Any changes in the configuration, we can change the code and run against the servers to reflect the changes.

Restoration is easy if something goes wrong. Auditing is also easy because we track each and every change to the configuration through code.

Q: What are ansible advantages over shell?

A: Ansible has multiple advantages over shell.

Idempotency: running shell scripts again and again may create duplicate resources and errors. Ansible has idempotency, it makes sure no duplicate resources are created even if we run playbook multiple times.

Scalability: Shell scripts have to run on servers individually. Ansible can connect multiple servers at the same time based on fork configuration.

Cross Platform: shell scripts can't be run against all Linux distributions. But Ansible can be heterogeneous, we can write playbooks to run against multiple Linux distributions.

Error handling: We have to check exit status to confirm the previous command is successful or not. Ansible takes care of this automatically.

Declared configuration: Ansible playbooks are easy to understand through YAML structure.

Rich Modules: Ansible has rich modules to interact with almost all popular platforms.

Ansible has a tag feature to control tasks and a vault to manage secrets.

Q: What are handlers in Ansible?

A: Ansible handlers are notifiers, when some task has a change happened, we can run another task through handlers.

For example, if one task changes the nginx configuration, it can notify another task called restarting nginx. Another example is when a systemctl service file changes we can notify other tasks to daemon reload and restart.

Q: What is the significance of the become keyword in Ansible playbooks?

A: become provides the sudo access to the playbook. It gives sudo access to every task mentioned in the playbook.

Q: How does Ansible handle idempotence, and why is it important in configuration management?

A: Meaning of idempotence is, the result should not change even if you run the script infinite times. It is highly important in configuration management because we run the scripts against servers multiple times, it should not create duplicates or errors.

Q: How do you handle errors and exceptions in Ansible playbooks?

A: When there is an error in ansible playbook, by default it will exit the program. But we can manage errors in Ansible too.

Ignore errors: this option allows the playbook to continue even if there is failure in a task.

Ignore_unreachable: We can ignore the task failure due to the host not reachable, it will continue further to run next tasks.

```
- name: This executes, fails, and the failure is ignored
  ansible.builtin.command: /bin/true
  ignore_unreachable: true
```

Failed_when: Ansible let us define what does it mean for the task failure, we can combine this with operators like ‘or’ ‘and’

```
- name: Fail task when the command error output prints FAILED
  ansible.builtin.command: /usr/bin/example-command -x -y -z
  register: command_result
  failed_when: "'FAILED' in command_result.stderr"
```

or based on the return code

```
- name: Fail task when both files are identical
  ansible.builtin.raw: diff foo/file1 bar/file2
  register: diff_cmd
  failed_when: diff_cmd.rc == 0 or diff_cmd.rc >= 2
```

Q: Describe the structure of an Ansible playbook.?

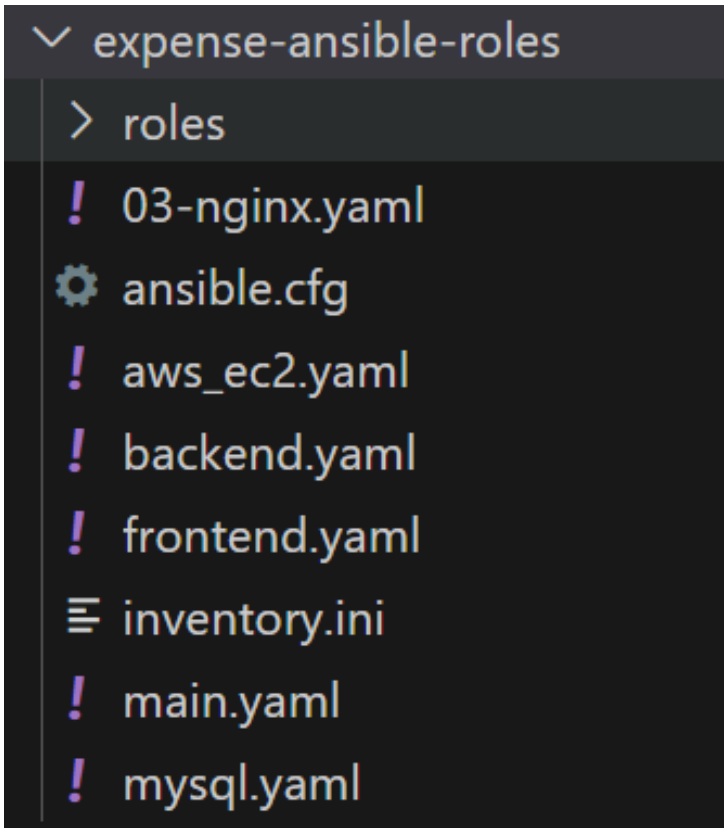
A: Ansible playbook follows YAML structure, it is a list of plays. It has the name of the play, hosts, become, vars, list of tasks.

```yaml
---
- name: Playbook to set up a web server
  hosts: web_servers          # Target group (or host) defined in inventory
  become: yes                 # Privilege escalation (sudo)
  vars:                       # Define any variables for the play
    web_root: /var/www/html
    web_service: httpd

  tasks:                      # Main list of tasks for the play
    - name: Install Apache web server
      ansible.builtin.yum:
        name: "{{ web_service }}"
        state: present

    - name: Copy index.html to web root
      ansible.builtin.copy:
        src: index.html
        dest: "{{ web_root }}/index.html"
      notify: restart apache   # Notify handler if this task changes the file
```

Q: How do you configure inventory files in ansible?

A: Inventory file is where we can manage the hosts, group of hosts and group of groups that ansible can manage. We use the .ini format. Inventory file can be specified under

We manage the inventory file in our playbook folder and mention it in ansible.cfg in the playbook folder.



We can mention the group level variables and host level variables in the inventory file too.

```
# Basic inventory file
[web_servers]
web1.example.com
web2.example.com
192.168.1.10

[db_servers]
db1.example.com
db2.example.com

[all_servers:children]  # Creating a group of groups
web_servers
db_servers

[all_servers:vars]     # Defining variables for a group
ansible_user=admin
ansible_ssh_private_key_file=~/.ssh/id_rsa
```

Q: How can you secure sensitive information, such as passwords, when working with Ansible?

A: We can use ansible vault feature to encrypt the sensitive data like usernames and passwords.

We can use ansible-vault command to make a simple yaml file as secured.

```
ansible-vault create secrets.yml
```

We can use the vault file as vars_files in the yaml file. While execution of playbook ansible asks for vault password.

```
# secrets.yml (encrypted with Ansible Vault)
db_password: "secure_password"
api_key: "secure_api_key"
```

```
---
- name: Example playbook using encrypted secrets
  hosts: all
  vars_files:
    - secrets.yml
  tasks:
    - name: Use the sensitive data
      ansible.builtin.shell: echo {{ db_password }}
```

```
ansible-playbook playbook.yml --ask-vault-pass
```