

Advanced Ansible Interview Questions

Author: [Zayan Ahmed](#) | Estimated Reading time: 5 min

Preparing for a role that requires expertise in Ansible? This guide compiles advanced Ansible interview questions to help you showcase your proficiency. Covering topics like roles, variables, templates, dynamic inventories, Ansible Vault, and best practices, these questions will test both your theoretical understanding and practical skills.



1. Core Concepts

Q1: What are Ansible roles, and how do they improve playbook organization?

- **Answer:** Roles are a way to organize Ansible playbooks into reusable and modular components. They separate tasks, variables, files, and templates into a standardized directory structure, making playbooks more maintainable and scalable.

Q2: Can you explain the difference between **vars, **defaults**, and **set_fact**?**

- **Answer:**
 - **vars:** Variables defined in playbooks, inventories, or roles.
 - **defaults:** Default variables for roles, overridden by other variable sources.

- **set_fact**: Sets variables dynamically during task execution, overriding other variable sources temporarily.

Q3: How does Ansible handle variable precedence?

- **Answer:** Ansible applies variable precedence based on a strict hierarchy, starting from role defaults (lowest) to extra vars (highest). Understanding this order ensures proper variable resolution in playbooks.
-

2. Advanced Playbook Techniques

Q4: How do you use **include_role** and **import_role**? What is the difference?

- **Answer:**
 - **include_role**: Dynamically includes a role at runtime, allowing for conditional execution.
 - **import_role**: Statically includes a role at parse time, making it part of the playbook structure.

Q5: What are some use cases for **block** and **rescue** in playbooks?

- **Answer:** **block** allows grouping tasks, while **rescue** provides error handling for tasks in the block. For example, using **block** for service deployment and **rescue** to revert changes if deployment fails.
-

3. Dynamic Inventories

Q6: What is a dynamic inventory, and how is it different from a static inventory?

- **Answer:**
 - **Dynamic Inventory**: Fetches host information from external sources (e.g., AWS, Azure, custom scripts) at runtime.
 - **Static Inventory**: Predefined in an inventory file (INI or YAML format).

Q7: Can you describe a scenario where you would use a dynamic inventory plugin?

- **Answer:** Using the AWS EC2 plugin to fetch instances dynamically based on tags or regions, ensuring playbooks adapt to infrastructure changes automatically.



ANSIBLE

4. Templates and Handlers

Q8: How do you use templates in Ansible?

- **Answer:** Templates (Jinja2 files) generate dynamic configuration files. Use the `template` module to copy templates to target hosts, substituting variables defined in playbooks or inventories.

Q9: What are handlers in Ansible, and how do they work?

- **Answer:** Handlers are tasks triggered by notifications from other tasks. For example, restarting a service when a configuration file changes.
-

5. Ansible Vault

Q10: What is Ansible Vault, and how do you manage sensitive data?

- **Answer:** Ansible Vault encrypts sensitive data like passwords and keys. Commands like `ansible-vault encrypt`, `decrypt`, and `edit` allow managing encrypted files.

Q11: Can you decrypt an Ansible Vault-encrypted file on the fly during execution?

- **Answer:** Yes, by providing the `--ask-vault-pass` or `--vault-password-file` option when running playbooks.
-

6. Custom Modules

Q12: How do you write a custom Ansible module?

- **Answer:** Custom modules are written in Python, following a specific structure. Use the `AnsibleModule` class for argument parsing and returning results.

Q13: Can you provide an example of a scenario where a custom module is required?

- **Answer:** When interacting with a proprietary API or performing tasks not supported by built-in modules.
-

7. Performance and Debugging

Q14: How do you optimize Ansible playbook performance?

- **Answer:**
 - Use `free strategy` for parallelism.
 - Avoid unnecessary `gather_facts`.
 - Cache facts using `fact_caching`.
 - Limit tasks to specific hosts with `when` conditions.

Q15: How do you debug an Ansible playbook?

- **Answer:** Use `-vvv` for verbose output, `debug` module to print variable values, and `--step` to execute tasks interactively.
-

8. Best Practices

Q16: What are some best practices for writing Ansible playbooks?

- **Answer:**
 - Organize playbooks using roles.
 - Use variables and templates to avoid hardcoding.
 - Encrypt sensitive data with Ansible Vault.
 - Write idempotent tasks to prevent unintended changes.

Q17: How do you test Ansible playbooks?

- **Answer:** Use tools like Molecule for unit testing roles and playbooks. Run playbooks in a staging environment before production.
-

9. Real-World Scenarios

Q18: How would you handle rolling updates using Ansible?

- **Answer:** Use the `serial` keyword to limit the number of hosts updated simultaneously, ensuring minimal downtime.

Q19: Describe how you would integrate Ansible with Jenkins.

- **Answer:** Use Jenkins to trigger Ansible playbooks via `ansible-playbook` command or Ansible plugins. Pass parameters like inventory files dynamically from Jenkins.
-

10. Miscellaneous

Q20: What is the purpose of the `meta` directory in roles?

- **Answer:** The `meta` directory contains metadata about the role, such as dependencies and author information.

Q21: How does Ansible handle dependencies between roles?

- **Answer:** Define role dependencies in the `meta/main.yml` file. Ansible ensures these roles are executed before the dependent role.
-

Mastering these advanced Ansible concepts and scenarios will not only prepare you for challenging interview questions but also empower you to handle complex automation tasks in real-world environments.

Follow me on [LinkedIn](#) for more 😊