

Great choice 👍 Ansible is another **core DevOps tool** and often comes up in interviews along with Terraform, Docker, and Kubernetes. Here's a structured set of **most frequently asked Ansible interview questions with answers**.

🚀 Ansible Interview Questions and Answers

****1. What is Ansible, and why is it used?****

****Answer:****

- * Ansible is an **open-source configuration management, provisioning, and orchestration tool**.
- * It uses **YAML-based playbooks**.
- * **Agentless** → doesn't need software installed on target machines, only SSH.
- * **Push-based** model (unlike Puppet/Chef, which are pull-based).

👉 Example use cases:

- * Installing packages
- * Configuring servers
- * Deploying applications
- * Managing multi-tier environments

****2. Difference between Ansible, Puppet, and Chef?****

****Answer:****

- * **Ansible** → Agentless, YAML, push model, easy learning curve.
- * **Puppet/Chef** → Require agents, DSL (domain-specific language), pull model, more complex.

👉 Most companies prefer Ansible due to simplicity.

****3. What are Playbooks, Roles, and Inventories in Ansible?****

****Answer:****

- * **Playbook** → A YAML file defining automation tasks.
- * **Role** → A way to organize playbooks into reusable units (tasks, vars, handlers, templates).
- * **Inventory** → File containing list of managed hosts.

👉 Example playbook:

```
``yaml
- hosts: webservers
  tasks:
    - name: Install Nginx
      apt:
        name: nginx
        state: present
...

```

****4. How does Ansible connect to managed nodes?****

****Answer:****

- * Uses ****SSH**** (Linux/Unix).
- * Uses ****WinRM**** (Windows).
- * No agents required on the nodes.

**5. What are Handlers in Ansible?**

****Answer:****

- * Handlers are tasks triggered only when ****notified**** by other tasks.
- * Example: restart service only if config file changes.

```yaml

tasks:

- name: Copy config file
 copy:
 src: nginx.conf
 dest: /etc/nginx/nginx.conf
 notify: Restart nginx

handlers:

- name: Restart nginx
 service:
 name: nginx
 state: restarted

```

**6. What is Idempotency in Ansible?**

****Answer:****

- * Idempotency → running a playbook multiple times produces the ****same result****.
- * Example: If a package is already installed, running `state: present` won't reinstall it.

**7. What is Ansible Galaxy?**

****Answer:****

- * A ****repository of pre-built roles**** shared by the community.
- * You can use it to speed up automation (e.g., installing Nginx, Docker).

```bash

ansible-galaxy install geerlingguy.nginx

```

**8. What are Ansible Facts?**

****Answer:****

- * Facts are ****system properties**** (IP, hostname, OS, CPU info) collected by Ansible at runtime.
- * Example:

```
```yaml
- debug:
 var: ansible_hostname
```
```

9. Difference between `vars`, `vars_files`, and `host_vars` in Ansible?

Answer:

- * **vars** → Inline variables inside playbook.
- * **vars_files** → External YAML files with variables.
- * **host_vars/group_vars** → Variables assigned to specific hosts or groups.

10. How do you secure sensitive data (passwords) in Ansible?

Answer:

- * Using **Ansible Vault**.

```
```bash
ansible-vault encrypt secrets.yml
ansible-playbook site.yml --ask-vault-pass
```
```

11. Can you explain Ansible Tower / AWX?

Answer:

- * **Ansible Tower (paid)** / **AWX (open-source)** → Web UI + REST API for Ansible.

- * Provides:

- * Role-based access
- * Centralized logs
- * Job scheduling
- * Visual dashboards

12. What are Tags in Ansible?

Answer:

- * Tags allow running **specific tasks** in a playbook.

```
```yaml
- name: Install Apache
 apt:
 name: apache2
 state: present
 tags: web
```
```

Run only tagged tasks:

```
```bash
ansible-playbook site.yml --tags web
```
```

13. What is the difference between `command`, `shell`, and `raw` modules?

****Answer:****

* ****command**** → Executes commands without shell.
* ****shell**** → Executes through a shell (can use pipes, redirects).
* ****raw**** → Runs command directly (useful for bootstrapping, e.g., installing Python on bare machines).

**14. What is the difference between `copy` and `template` modules?

****Answer:****

* ****copy**** → Copies static files.
* ****template**** → Uses Jinja2 templating (dynamic files).

Example template:

```
``yaml
- name: Create config
  template:
    src: nginx.conf.j2
    dest: /etc/nginx/nginx.conf
````
```

---

## \*\*15. How do you test Ansible playbooks before running on production?

**\*\*Answer:\*\***

\* Use **\*\*`--check` mode\*\*** (dry run).

```
``bash
ansible-playbook site.yml --check
````
```

* Use ****staging inventory**** first.
* Use ****Molecule**** for testing roles.

🚀 Scenario-Based Ansible Questions

1. ****Your playbook fails because Python is missing on a remote server. How do you fix it?****

👉 Use ****`raw` module** to install Python.

2. ****You want to apply a playbook only on 2 out of 10 servers. How?**

👉 Use inventory groups or ****`--limit` flag**.

```
``bash
ansible-playbook site.yml --limit "web1,web2"
````
```

3. **\*\*You need to ensure service restarts only when the config changes. How?**

👉 Use **\*\*handlers with notify\*\***.

4. **\*\*You want to deploy one task to RedHat servers and another to Ubuntu**

servers. How?

👉 Use conditionals with `ansible\_os\_family`.

```
```yaml
- name: Install Apache on RHEL
  yum:
    name: httpd
    state: present
  when: ansible_os_family == "RedHat"
```
```

---

✅ With these Q&As, you'll be confident in both **theory + real-world scenarios**.

👉 Do you want me to now create a **mock interview drill** (interviewer asks, you answer live) for **Ansible + Terraform combined**, so you get **real practice**?

-----  
-----  
-----

Perfect 👍 Here's a **complete set of 10 most-asked Ansible + Terraform interview questions** with clear, structured answers.  
You can memorize these as talking points and adapt them to your own experience.

---

# 🧐💻 Ansible + Terraform Interview Q&A

---

### ? Q1. Can you explain the difference between Ansible and Terraform? When would you use one over the other?

**\*\*Answer:\*\***

\* **\*\*Terraform\*\*** → Infrastructure provisioning (VMs, networks, storage, load balancers, etc.).

\* **\*\*Ansible\*\*** → Configuration management (installing software, applying configs, managing services).

\* Terraform creates the servers, Ansible configures what runs on them.

👉 Example: Use Terraform to provision 5 EC2 instances in AWS, then use Ansible to install Nginx on them.

---

### ? Q2. Terraform is declarative and Ansible is procedural – what does that mean in practice?

**\*\*Answer:\*\***

\* **\*\*Declarative (Terraform):\*\*** You define the *desired state* ('I want 3 EC2 instances') and Terraform ensures reality matches it.

\* **\*\*Procedural (Ansible):\*\*** You define the *steps* to get there (install package, copy config, start service).

👉 Example: In Terraform, you say "3 servers." In Ansible, you say "install Apache → copy config → restart Apache."

---

### ? Q3. How does Terraform maintain the state of resources, and why is

`terraform.tfstate` important? What happens if it gets corrupted?

**\*\*Answer:\*\***

- \* Terraform stores resource details in a **\*\*state file\*\*** (`terraform.tfstate`).
- \* State tracks what exists in the real environment vs. what's in your code.
- \* If it's corrupted or deleted → Terraform can't track resources properly (may try to recreate).

👉 Best practice: Store state remotely (S3 + DynamoDB lock, Terraform Cloud, etc.) and enable backups.

---

### ? Q4. In Terraform, what is the difference between **\*\*resource\*\***, **\*\*data\*\***, and **\*\*module\*\*** blocks? Can you give examples?

**\*\*Answer:\*\***

- \* **\*\*Resource:\*\*** Creates infrastructure.

```
```hcl
resource "aws_instance" "web" { ... }
```
```

- \* **\*\*Data:\*\*** Reads existing resources.

```
```hcl
data "aws_vpc" "default" { default = true }
```
```

- \* **\*\*Module:\*\*** Reusable code bundle.

```
```hcl
module "network" { source = "../vpc-module" }
```
```

---

### ? Q5. In Ansible, how do you ensure that a configuration change triggers a restart of a service only if necessary?

**\*\*Answer:\*\***

By using **\*\*handlers\*\***.

```
```yaml
tasks:
  - name: Copy nginx config
    copy:
      src: nginx.conf
      dest: /etc/nginx/nginx.conf
      notify: restart nginx
```
```

```
handlers:
 - name: restart nginx
 service:
 name: nginx
 state: restarted
```
```

👉 Restart happens only when the config actually changes.

? Q6. Suppose you have 100 servers, and you want to apply an Ansible playbook only to 10 specific ones without changing inventory. How do you do that?

****Answer:****
Use `--limit`.

```
```bash
ansible-playbook site.yml --limit "server1,server2,server3"
```
```

👉 This restricts execution to specific hosts without modifying inventory.

? Q7. Your Terraform plan shows changes that you did not expect. How do you debug and troubleshoot such issues?

****Answer:****

1. Run `terraform plan` → inspect details.
 2. Check drift: resources may have been modified outside Terraform.
 3. Run `terraform refresh` or `terraform import` to align state.
 4. Review `lifecycle` settings (e.g., `ignore_changes`).
- 👉 Example: If an EC2 tag was changed manually in AWS, Terraform will detect and plan to change it back.

? Q8. How do you handle secrets in both Ansible and Terraform?

****Answer:****

* ****Ansible:**** Use ****Ansible Vault**** to encrypt sensitive files (`ansible-vault encrypt secrets.yml`).

* ****Terraform:**** Use ****variables + external secret stores**** (AWS Secrets Manager, Vault, SSM Parameter Store). Never hardcode secrets in code or state files.

? Q9. Scenario: You want to create AWS infrastructure with Terraform, and then configure the instances with Ansible. How would you integrate both?

****Answer:****

- * Step 1: Use Terraform to provision EC2 instances.
 - * Step 2: Use Terraform outputs (IP addresses) to generate an Ansible inventory file.
 - * Step 3: Run Ansible playbooks on those hosts.
- 👉 Integration example:

```
```hcl
output "ec2_ips" {
 value = aws_instance.web.*.public_ip
}
```
```

Then pass that to Ansible dynamic inventory.

? Q10. What are some best practices you follow when writing Ansible playbooks and Terraform code?

****Answer:****

* **Terraform Best Practices:**

- * Store state remotely & lock it.
- * Use modules for reusability.
- * Follow naming conventions.
- * Run `terraform fmt` + `terraform validate`.

* **Ansible Best Practices:**

- * Keep playbooks idempotent.
- * Use roles for reusability.
- * Encrypt secrets with Vault.
- * Test with `--check` (dry run) before applying changes.

✅ With these, you can comfortably handle **80-90% of real interview questions** on Ansible + Terraform.

Would you like me to also prepare a **scenario-based Q&A set** (like “what if Terraform state is lost?” or “how do you debug a failed Ansible playbook on 1 of 100 servers?”) – those are **very common follow-ups** in real interviews.

Great 🧨 let's go deeper into **real-world, scenario-based interview questions** for **Ansible + Terraform**.

These are the type of “What would you do if...” questions interviewers love because they test **practical problem-solving** beyond just theory.

⚡ Scenario-Based Ansible + Terraform Questions & Answers

🤔 Scenario 1:

You ran `terraform apply`, but it failed halfway. Now some resources are created, some are not. What would you do?

****Answer:****

- * First, run `terraform plan` to see the current state vs. code.
- * If state is mismatched, run `terraform refresh`.
- * If some resources exist outside state, import them using `terraform import`.
- * Then re-run `terraform apply` safely.

👉 Key Point: Never manually delete resources unless you're sure – always fix state first.

🤔 Scenario 2:

Your Terraform state file got corrupted or accidentally deleted. How do you recover?

****Answer:****

- * If using **remote backend (S3, Terraform Cloud, etc.)**, restore from last backup.
- * If no backup exists, re-import existing resources with `terraform import`.

* Worst case: manually clean environment and re-apply code.
👉 Best practice: Always enable state locking and versioning in backend (e.g., S3 + DynamoDB).

? Scenario 3:

You applied an Ansible playbook on 100 servers, and 1 server failed. What would you do?

****Answer:****

- * Check the error logs (`-vvv` for verbose mode).
- * Use `--limit` to re-run only on that host.
- * If it's a transient error, run with `--start-at-task` to continue from the failed task.

👉 This avoids re-running everything unnecessarily.

? Scenario 4:

You want to provision infrastructure with Terraform and configure it with Ansible, but the public IPs of instances are not known before creation. How do you solve this?

****Answer:****

- * Use ****Terraform outputs**** to capture IPs.
- * Pass outputs into Ansible dynamic inventory (e.g., via script or plugin).

👉 Example:

```
```hcl
output "ec2_ips" {
 value = aws_instance.web.*.public_ip
}
```
```

Then generate inventory from that output.

? Scenario 5:

A teammate edited AWS infrastructure manually outside of Terraform. Next `terraform plan` shows lots of unwanted changes. How do you handle this drift?

****Answer:****

- * Run `terraform plan` to identify drift.
- * If changes are acceptable → run `terraform import` or update code.
- * If not acceptable → apply Terraform to override manual changes.

👉 Best practice: Educate team → ****never change infra manually**** when Terraform is the source of truth.

? Scenario 6:

You want to run an Ansible playbook in production, but you're worried it might break something. What do you do?

****Answer:****

- * Run in **check mode** (`--check`) to simulate changes without applying.
- * Use `--diff` to preview file changes.
- * Test in staging first.
- 👉 This prevents outages due to mistakes.

? Scenario 7:

Your Ansible playbook takes too long because it runs tasks sequentially across 500 servers. How do you optimize?

Answer:

- * Increase **forks**:

```
```bash
ansible-playbook site.yml -f 50
```
```

- * Use **tags** to run only required tasks.
- * Cache facts to avoid re-gathering.
- 👉 Parallelism + selective execution reduces runtime.

? Scenario 8:

In Terraform, how do you ensure a resource (e.g., S3 bucket) is never accidentally destroyed?

Answer:

Use `lifecycle` block:

```
```hcl
resource "aws_s3_bucket" "data" {
 bucket = "my-data-bucket"
 lifecycle {
 prevent_destroy = true
 }
}
```
```

👉 Even if someone runs `terraform destroy`, it won't delete that resource.

? Scenario 9:

Ansible Vault is used to encrypt secrets, but someone needs to run playbooks without vault password. How do you handle this securely?

Answer:

- * Don't share the vault password directly.
- * Use Ansible Vault with CI/CD pipeline (e.g., Jenkins, GitHub Actions) where vault key is injected securely via secrets manager.
- * For individuals, use `ansible-vault view` to share only decrypted values when necessary.

? Scenario 10:

Your Terraform codebase has grown huge with many environments (dev, stage, prod). How do you structure it?

****Answer:****

- * Use ****modules**** for reusable components (e.g., VPC, EC2, RDS).
- * Separate environment directories:

```
```\n/environments/dev\n/environments/stage\n/environments/prod\n```
```

- \* Use workspaces or different state backends.  
👉 This avoids confusion and enforces separation of concerns.

---

✅ These scenarios show interviewers you can **\*\*debug, troubleshoot, and apply best practices in real-world DevOps situations\*\***.

Would you like me to now create a **\*\*crisp 1-page cheat sheet\*\*** (bullet style) summarizing all **\*\*Terraform + Ansible interview answers\*\*** so you can revise quickly before an interview?