# DevOps — Ansible (25 Questions)

---

### Q1: You run an Ansible playbook to provision EC2 instances, but some tasks hang indefinitely on SSH.

**Answer:**
This can happen if the new EC2 instances aren't ready for SSH yet or have restrictive SG rules. Add a wait_for task to ensure port 22 is open before gathering facts, and verify ansible_user matches the AMI's default.

**Sample Points:**

- Use `wait_for` before `gather_facts`.
- Ensure SG allows SSH from control node.
- Use correct remote user for AMI type.

**Example Code:**

```
- name: Wait for SSH
  wait_for:
    host: "{{ inventory_hostname }}"
    port: 22 delay: 5 timeout: 300
```



---

### Q2: Your Ansible run modifies files every time even though the contents haven't changed.

**Answer:**
The task is not idempotent — for file edits, use `lineinfile`/ `blockinfile`with proper regex, and for templates, use template which only updates if checksums differ.

**Sample Points:**

- Idempotency avoids unnecessary changes.

- Use correct module for text changes.
- Compare generated vs. target file.

**Example Code:**

```
- name: Ensure line present
  lineinfile:
    path: /etc/sysctl.conf
    regexp: '^net.ipv4.ip_forward'
    line: 'net.ipv4.ip_forward=1'
```

---

## Q3: A task needs to run only when a file exists on the target host.

**Answer:**

Use `stat` module to check and conditionally run the next task with `when`.

**Sample Points:**

- `stat` returns `exists` flag.
- Avoid failing on missing files.
- Clean conditional syntax.

**Example Code:**

```
 - stat: path=/etc/my.conf

  register: conf_file

- name: Do something
command: cat /etc/my.conf
 when: conf_file.stat.exists
```

---

## Q4: Playbook fails on some hosts due to Python not being installed.

**Answer:**

Use `raw` module to install Python first, as Ansible modules need Python.

**Sample Points:**

- Cloud images (minimal) may lack Python.
- raw bypasses Python requirement.
- Install python3 early in play.

**Example Code:**

```
- name: Install Python
  raw: sudo apt-get update && sudo apt-get install -y python3
```

---

### Q5: You want to encrypt sensitive variables in your repo.

**Answer:**
Use `ansible-vault encrypt` for the vars file, and store the vault password outside VCS.
**Sample Points:**

- Never commit plain secrets.
- Vault key in CI/CD via secret store.
- Can encrypt single vars or whole files.

**Example Code:**

```
ansible-vault encrypt group_vars/prod/secrets.yml
```

---

### Q6: You need to run a specific set of tasks only on RHEL-based hosts.

**Answer:**
Use `when: ansible_facts['os_family'] == 'RedHat'`.
**Sample Points:**

- Use gathered facts for OS-specific logic.
- Avoid hardcoding hostnames.
- Keep tasks portable with conditionals.

**Example Code:**

```
when: ansible_facts['os_family'] == "RedHat"
```

---

### Q7: Playbook must be rerun without re-executing heavy install tasks.

**Answer:**

Use `creates` param in `command` or register a state flag file.

**Sample Points:**

- Skips task if output already exists.
- Avoids redundant installs.
- Faster reruns.

**Example Code:**

```
- name: Install app
  command: /opt/install.sh creates=/opt/app_installed.flag
```

---

### Q8: You want to dynamically pull inventory from AWS EC2.

**Answer:**

Use the `aws_ec2` dynamic inventory plugin and configure AWS credentials in env/credentials file.

**SamplePoints:**

- No need for static host files.
- Tag filtering for host groups.
- Refresh inventory automatically.

**Example Code:**

```
plugin: aws_ec2
regions:
  - ap-south-1
keyed_groups:
```

```
  -key: tags.Name
```

---

### Q9: Need to reuse task logic across multiple playbooks.

**Answer:**

Use roles to package tasks, vars, handlers, and templates together.

**Sample Points:**

- Roles promote reusability.
- Avoids duplicating tasks.
- Makes plays cleaner.

**Example Code:**

```
- hosts: web
  roles:
    - nginx_setup
```

---

### Q10: Playbook must read a variable from another host in the same play.

**Answer:**

Use `hostvars` to access facts/vars from other hosts.

**Sample Points:**

- hostvars is a dictionary of all hosts.
- Requires both hosts in same play context.
- Useful for leader-worker configs.

**Example Code:**

```
db_host: "{{ hostvars['db1'].ansible_host }}"
```

---

### Q11: Task needs to run only if a service is not already running.

**Answer:**
Check with `service_facts` and conditionally start.
**Sample Points:**

- Avoids restarting running services.
- service_facts gathers all services.
- Improves idempotency.
  **Example Code:**

```
- service_facts:
- service:
    name: nginx
    state: started
  when: "'nginx' not in services or services['nginx'].state !=
'running'"
```

---

## Q12: Playbook execution must stop if a critical task fails.

**Answer:**
Use `any_errors_fatal: true` at play level.
**Sample Points:**

- Ensures all hosts stop on failure.
- Useful for critical infra changes.
- Avoid partial config states.
  **Example Code:**

```
- hosts: all
  any_errors_fatal: true
```

---

## Q13: Need to set variables that are evaluated only at execution time.

**Answer:**
Use `set_fact` with Jinja templates.
**Sample Points:**

- set_fact is dynamic at runtime.
- Useful for computed values.
- Facts persist for rest of play.

**Example Code:**

```
- set_fact:
    backup_path: "/backups/{{ inventory_hostname }}/{{
ansible_date_time.date }}"
```

---

## Q14: Playbook fails on missing variable in a template.

**Answer:**
Set `jinja2_native=True` and `default` filter in template to avoid undefined errors.
**Sample Points:**

- default avoids undefined var crash.
- Set safe defaults.
- Reduces fragile templates.

**Example Code:**

```
{{ some_var | default('N/A') }}
```

---

## Q15: Need to run some tasks as another Linux user without switching SSH login.

**Answer:**
Use `become` with `become_user`.
**Sample Points:**

- become allows privilege escalation.
- Works for sudo or su target.
- Avoids separate SSH creds.

**Example Code:**

```
- name: Run as postgres
  command: psql -c "SELECT 1"
  become: true
  become_user: postgres
```

---

## Q16: Deploying app needs different config files per environment.

**Answer:**
Use group_vars for environment-specific vars.

**Sample Points:**

- Separate vars by inventory group.
- Avoids conditionals inside playbooks.
- Cleaner separation of configs.

**Example Code:**

```
group_vars/
prod.yml
dev.yml
```

---

## Q17: Playbook should stop at a certain task for debugging.

**Answer:**
Use `meta: end_play` or `pause` for interactive debug.

**Sample Points:**

- end_play stops entire play.

- pause allows manual checks.
- Useful in staging/testing.

**Example Code:**

```
- meta: end_play
```

---

## Q18: Need to install packages on both Debian and RHEL hosts with one task.

**Answer:**

Use `package` module with variables mapping per OS.

**Sample Points:**

- package is generic across distros.
- Avoids duplicate tasks per OS.
- Use var mapping for names.

**Example Code:**

```
vars:
  pkg_name:
    RedHat: httpd
    Debian: apache2
- package:
    name: "{{ pkg_name[ansible_os_family] }}"
    state: present
```

---

## Q19: A handler is not running even though task notifies it.

**Answer:**

Handlers run at the end of plays unless `meta: flush_handlers` is used.

**Sample Points:**

- flush_handlers triggers early.

- Handlers run once per play.
- Useful for service restarts mid-play.

**Example Code:**

```
- meta: flush_handlers
```

---

## Q20: Need to securely fetch secrets from AWS SSM in playbook.

**Answer:**
Use `aws_ssm` lookup plugin with IAM role permissions.
**Sample Points:**

- Avoids hardcoded creds.
- IAM role least-privilege.
- Pull secrets at runtime.

**Example Code:**

```
db_pass: "{{ lookup('aws_ssm', '/prod/db_pass', region='us-east-1')
}}"
```

---

## Q21: Limit playbook run to a subset of tasks for quick testing.

**Answer:**
Use `--tags` and `--skip-tags` with well-tagged tasks.
**Sample Points:**

- Tags speed up testing.
- Skip irrelevant tasks.
- Tag logically by function.

**Example Code:**

```
ansible-playbook site.yml --tags "nginx,deploy"
```

---

### Q22: Playbook fails due to SSH host key verification prompt.

**Answer:**

Set `host_key_checking = False` in ansible.cfg for automation (with caution).

**Sample Points:**

- Disables fingerprint prompt.
- Security trade-off: trust host blindly.
- Prefer adding host key to known_hosts.

**Example Code:**

```
[defaults]
host_key_checking = False
```

---

### Q23: Need to run a task only on first host in a group.

**Answer:**

Use `when: inventory_hostname == groups['web'][0]`.

**Sample Points:**

- Controls execution scope.
- Avoids duplicate operations.
- Index 0 is first host.

**Example Code:**

```
when: inventory_hostname == groups['web'][0]
```

---

### Q24: A variable value needs to be computed using output from a previous command.

**Answer:**

Register command output and use `set_fact`.

**Sample Points:**

- Register stores task output.
- Use stdout_lines for list processing.
- set_fact makes it available globally.

**Example Code:**

```
- command: hostname
  register: host_out
- set_fact:
    fqdn: "{{ host_out.stdout }}.example.com"
```

---

### Q25: Need to ensure a directory exists with specific permissions.

**Answer:**

Use file module with `state: directory`.

**Sample Points:**

- Ensures idempotent creation.
- Sets ownership and mode.
- Works for nested paths.

**Example Code:**

```
- file:
    path: /opt/data
    state: directory
    owner: appuser
    mode: '0750'
```