Ansible Interview & Real-World Project Playbook

The document is a professional Ansible interview and project playbook **covering architecture, playbooks, roles, and real-world automation scenarios.**
It includes advanced topics like CI/CD integration, GitOps workflows, Molecule testing, and hybrid cloud automation **(AWS, Kubernetes, Terraform).**

☑ 1. Basics & Architecture

Q1: Explain Ansible architecture.

Answer:

- Agentless automation using SSH (default) or WinRM (Windows).
- Control Node: Executes playbooks.
- Managed Nodes: Target machines.
- Inventory: List of target nodes.
- Modules: Actionable code run on target nodes.
- Plugins: Extend functionality (callbacks, connections, etc.).
- Playbooks: YAML describing automation workflow.

🔥 **Real-Time:** Automated 100+ EC2 instance configurations using Ansible over SSH with PEM keys, no agent installation required.

☑ Q2: Where does Ansible fit

Scenario: Already have Terraform, AWS infra, Kubernetes clusters, Docker, GitLab CI, microservices, and Shell scripts running**. Where does Ansible fit, and what are the best use-cases/tasks for Ansible in this mature ecosystem?**

☑ 2.1. Post-Provisioning OS Configuration (Terraform → Ansible)

Use Case:
After Terraform provisions infra, Ansible handles:

- Install base OS packages (curl, unzip, python3)
- Configure sysctl, limits.conf, firewalld

- Harden SSH configs
- Mount EBS volumes
- Cloud-init alternative

🔥 **Real-Time:** Post Terraform, Ansible ensures EC2 nodes are production-ready for Kubernetes join.

```
- name: Install base packages
  apt:
    name:
      - curl
      - unzip
      - python3
    state: present
```

☑ 2.2. GitLab CI / Runners Automation

Use Case:

- Install and register self-hosted GitLab Runners
- Auto-update binaries
- Manage runner tags, jobs, and executors

🔥 **Real-Time:**
Automated runner scaling in AWS for CI pipelines.

```
- name: Register GitLab Runner
  shell: >
    gitlab-runner register --non-interactive
    --url https://gitlab.example.com/
    --registration-token "{{ runner_token }}"
    --executor docker
    --docker-image alpine:latest
```

☑ 2.3. Kubernetes Backup with Velero

Use Case:
Automate Kubernetes cluster backup.

🔥 **Real-Time:** On-demand Velero backups before major updates.

```
- name: Trigger Velero backup
  hosts: localhost
  tasks:
    - name: Create Velero backup
      shell: velero backup create my-backup --include-namespaces microservices
```

Q2.4: Difference between Ansible ad-hoc commands and playbooks?

Answer:

- **Ad-hoc**: Quick, one-off tasks.
- **Playbook**: Reusable YAML automation.

Ad-hoc Example:

```
ansible all -m ping -i inventory.ini
```

Playbook Example:

```
- name: Install nginx
  hosts: web
  tasks:
    - name: Install nginx
      apt:
        name: nginx
        state: present
```

☑ 2. Playbooks, Roles & Structure

Q3: Explain Ansible roles and real-world usage.

Answer:

- Modular, reusable units with standard folder structure.
- Tasks, vars, templates, handlers, defaults.
- Easy to share across projects.

🔥 **Real-Time:** Standardized NGINX role used across Dev, QA, and Prod for uniform web server setup.

```
- hosts: web
  roles:
    - nginx
```

Q4: How do you use variables in Ansible?

Answer:

- Multiple locations: playbooks, roles, inventory, CLI.
- Precedence: Extra vars > Playbook vars > Role defaults.

Example:

```
vars:
  package_name: nginx
tasks:
  - name: Install package
```

```
    apt:
      name: "{{ package_name }}"
      state: present
```

☑ 3. Real-Time Scenarios

Q5: How do you handle secrets in Ansible?

Answer:

- Use **Ansible Vault** for encrypting sensitive data.

🔥 **Real-Time:** Encrypted AWS keys for CI/CD usage with GitLab CI.

Vault Encrypt:

```
ansible-vault encrypt secrets.yml
```

Usage in playbook:

```
vars_files:
  - secrets.yml
```

Q6: How do you manage AWS dynamic infrastructure in Ansible?

Answer:

- Use **dynamic inventory** plugins like aws_ec2.

🔥 **Real-Time:** Auto-discovered EC2 instances based on tags for deployments.

Command:

```
ansible-inventory -i aws_ec2.yml --graph
```

☑ 4. Troubleshooting & Debugging

Q7: How to debug an Ansible playbook?

Answer:

- Use verbosity: -vvv
- Use debug module.

Example:

```
- debug:
    msg: "Variable value: {{ variable_name }}"
```

Q8: What to do when Ansible SSH fails?

Answer:

- Check SSH keys, user, inventory format.
- Test:

```
ansible all -m ping -i inventory.ini -u ec2-user --private-key=key.pem
```

☑ 5. Advanced Topics

Q9: Explain idempotence in Ansible.

Answer:

- Multiple playbook runs achieve the same state.
- Use stateful modules.

🔥 **Real-Time:** CI pipeline applies daily OS hardening with no redundant changes.

Example:

```
- name: Ensure nginx is installed
  apt:
    name: nginx
    state: present
```

Q10: Can you create reusable Ansible Galaxy roles?

Answer:

- Yes, using:
- `ansible-galaxy init my_role`
- Share via Galaxy or internal Git.

Q11: Explain handlers with real-world use.

Answer:

- Triggered tasks that run only when notified.

🔥 **Real-Time:**
Restart Nginx only if the configuration changes.

```
tasks:
```

```
  - name: Update nginx config
    template:
      src: nginx.conf.j2
      dest: /etc/nginx/nginx.conf
    notify: restart nginx

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

Q12: Error handling in playbooks?

Answer:

- Use block, rescue, always.

🔥 **Real-Time:** Handled AWS rate-limit errors gracefully.

```
tasks:
  - block:
      - name: Install nginx
        apt:
          name: nginx
          state: present
    rescue:
      - debug:
          msg: "Installation failed"
    always:
      - debug:
          msg: "Task complete"
```

☑ 6. CI/CD Integration

Q13: Ansible in GitLab CI/CD pipeline?

Answer:
Automated infra + app deployment.

```
deploy:
  stage: deploy
  script:
    - ansible-playbook -i inventory.ini deploy.yml
```

☑ 7. Cloud & Kubernetes Integration

Q14: Provision AWS infrastructure?

Answer:

- Use AWS modules.

```
- name: Create EC2
  amazon.aws.ec2_instance:
    name: "WebServer"
    image_id: "ami-0abcdef1234567890"
    instance_type: "t2.micro"
```

Q15: Manage Kubernetes resources?

Answer:

- Use kubernetes.core.k8s module.

🔥 **Real-Time:**
Automated YAML deployments.

```
- name: Deploy Nginx
  kubernetes.core.k8s:
    state: present
    definition: "{{ lookup('file', 'nginx-deployment.yml') }}"
```

☑ 8. Ansible + Terraform (IaC Combo)

Q16: How do you integrate Terraform with Ansible?

Answer:

- Terraform provisions infra.
- Terraform outputs EC2 IPs.
- Ansible configures infra post-provision.

🔥 **Real-Time:** Terraform outputs IPs, Ansible installs Docker and Kubernetes tooling.

Terraform:

```
output "ec2_instance_ips" {
  value = aws_instance.web.*.public_ip
}
```

Ansible Inventory:

```
[web]
ec2-xx-xxx-xx-xxx.compute-1.amazonaws.com
```

CI/CD Example:

```
deploy:
  stage: deploy
  script:
    - terraform apply -auto-approve
    - ansible-playbook -i inventory/terraform_output.ini playbooks/site.yml_
```

☑ 9. Ansible Collections

Q17: What are Ansible collections?

Answer:

- Packaged roles, modules, plugins.

Usage:

```
ansible-galaxy collection install amazon.aws
```

Playbook:

```
- name: Launch EC2
  amazon.aws.ec2_instance:
    name: "DemoInstance"
    instance_type: "t2.micro"
    image_id: "ami-xyz"
```

🔥 **Real-Time:** Used amazon.aws collection for updated AWS modules.

☑ 10. Testing with Molecule

Q18: How do you test roles before production?

Answer:

- Use Molecule with Docker driver.
- Integrate Molecule tests in CI.

🔥 **Real-Time:** Roles tested in GitLab CI before merging.

Command:

```
molecule test
```

GitLab CI:

```
test_role:
  stage: test
  image: python:3.9
```

```
  script:
    - pip install molecule[docker] ansible
    - molecule test
```

☑ 11. Performance Optimisation

Q19: How do you optimise playbooks for large environments?

Answer:

- Increase forks.
- Enable pipelining.
- Use async for long tasks.
- Disable fact gathering if not needed.

🔥 **Real-Time:** 300+ EC2 patching time reduced from 40 mins to 15 mins.

Config:

```
[defaults]
forks = 50
gathering = smart

[ssh_connection]
pipelining = True
```

Async Example:

```
- name: Run in background
  command: /path/to/script.sh
  async: 600
  poll: 0
```

☑ 12. Hybrid Cloud Management

Q20: How to manage hybrid cloud environments?

Answer:

- Use dynamic inventory and cloud collections.
- Manage AWS, Azure, On-prem, Kubernetes.

🔥 **Real-Time:**
Multi-cloud (AWS + Azure + On-prem) automation for microservices.

Inventory:

```
[aws]
aws-instance-1
```

```
[k8s]
k8s-node-1

[onprem]
onprem-server-1
```

☑ Ansible Command Cheat Sheet (Quick Reference)

Here's your professional **top commands list** for fast review.

Command	Description
ansible --version	Check Ansible version
ansible all -m ping -i inventory.ini	Test connectivity
ansible-playbook -i inventory/prod.ini playbook.yml	Run playbook
ansible-vault encrypt secrets.yml	Encrypt secrets
ansible-vault view secrets.yml	View encrypted secrets
ansible-galaxy install collection_name	Install collection
ansible-lint playbook.yml	Lint playbook
molecule test	Test roles with Molecule
ansible-inventory --graph -i aws_ec2.yml	Visualise inventory
ansible-pull -U https://repo.git site.yml	Run pull mode

☑ Advanced Troubleshooting Playbook (Professional Reference)

Issue	Solution
SSH connection failure	Verify key, user, and security groups. Use -vvvv for details.
Playbook hangs	Increase forks, enable pipelining in ansible.cfg.
AWS rate limit	Add retries or use block/rescue for handling API errors.
Playbook not idempotent	Add creates, removes, or use when conditions.
Slow execution	Disable facts (gather_facts: false), use async tasks.
Vault password mismatch	Ensure correct --vault-id or store in CI/CD secrets.
YAML syntax errors	Use ansible-lint, validate with yamllint.
Debug specific task	Use --start-at-task="Task Name" or --step.

🔥 Real-Time: During AWS multi-region rollout, SSH failures were debugged using:

```
ansible-playbook -vvvv -i inventory/aws_ec2.yml playbooks/deploy-app.yml
```

☑ 14. GitOps Workflow with Ansible

Q21: How do you integrate Ansible in a GitOps workflow?

Answer:

- Use Git as the single source of truth.
- Trigger playbooks via GitLab CI or GitHub Actions on commit.
- Manage environments via branch strategy (develop, staging, production).

🔥 **Real-Time:** Used GitLab CI to trigger Ansible deployments when the main branch was updated

Example GitLab CI:

```
stages:
  - deploy

deploy_to_dev:
  stage: deploy
  script:
    - ansible-playbook -i inventory/dev.ini playbooks/site.yml
  only:
    - develop

deploy_to_prod:
  stage: deploy
  script:
    - ansible-playbook -i inventory/prod.ini playbooks/site.yml
  only:
    - main
```

☑ **15. Migrating from Shell Scripts to Ansible**

Q22: How do you migrate legacy shell scripts to Ansible playbooks?

Answer:

- Break shell scripts into tasks.
- Replace imperative commands with idempotent Ansible modules.
- Incrementally test conversions.
- Modularise into roles.

🔥 **Real-Time:** Migrated 1,000+ lines of provisioning scripts into reusable roles.

Shell Script:

```
apt update
apt install -y nginx
systemctl start nginx
```

Ansible Playbook:

```yaml
- hosts: web
  tasks:
    - name: Update apt cache
      apt:
        update_cache: yes

    - name: Install nginx
      apt:
        name: nginx
        state: present

    - name: Start nginx service
      service:
        name: nginx
        state: started
```

☑ 16. Advanced Troubleshooting

Q23: How to debug slow Ansible runs?

Answer:

- Increase forks in ansible.cfg.
- Enable pipelining.
- Use -vvv verbosity.
- Disable fact gathering if not needed.

🔥 **Real-Time:** Enabled pipelining and raised forks to 50 for faster EC2 updates.

Config:

```
[defaults]
forks = 50
gathering = smart

[ssh_connection]
pipelining = True
```

Q24: How to handle non-idempotent tasks?

Answer:

- Use creates, removes, or conditionals.

🔥 **Real-Time:** DB migrations were skipped after the first run using creates.

Example:

```yaml
- name: Run migration script
  command: ./migrate.sh
  args:
    creates: /var/lib/app/migration_done.flag
```

☑ 19. GitHub-Ready Project Structure

🔥 **Real-Time:**
Used this structure in client automation projects — roles, inventories, and pipelines cleanly organised for maintainability.

```
ansible-project/
├── ansible.cfg
├── inventory/
│   ├── dev.ini
│   └── prod.ini
├── playbooks/
│   ├── site.yml
│   └── webserver.yml
├── roles/
│   ├── nginx/
│   │   ├── tasks/
│   │   │   └── main.yml
│   │   ├── templates/
│   │   │   └── nginx.conf.j2
│   │   └── handlers/
│   │       └── main.yml
├── vault/
│   ├── dev_secrets.yml
│   └── prod_secrets.yml
├── molecule/
│   └── default/
│       ├── converge.yml
│       └── molecule.yml
├── .gitlab-ci.yml
├── .github/workflows/ansible.yml
└── README.md
```