# ANSIBLE PRACTICAL DEMONSTRATION - COMPLETE GUIDE

---

## PREREQUISITES

**What you need:**

1. Two EC2 instances on AWS
2. MobaXterm or similar SSH client
3. Basic Linux command knowledge

**Server names in this example:**

- Server 1: shell-script-example (Ansible control node)
- Server 2: test (Target/managed node)

---

## PART 1: INSTALLATION AND SETUP

---

### STEP 1: INSTALL ANSIBLE ON CONTROL NODE

**Login to shell-script-example server**

**Update system packages:**

sudo apt update

**Why this is important:** Run this command every time a new instance is created for the first time. This updates the package list to get latest versions available.

**Install Ansible:**

sudo apt install ansible -y

**Verify installation:**

ansible --version

**Expected output:**

ansible [core 2.12.0]

config file = /etc/ansible/ansible.cfg

configured module search path = ['/home/ubuntu/.ansible/plugins/modules']

ansible python module location = /usr/lib/python3/dist-packages/ansible

python version = 3.10.6

---

**STEP 2: SETUP PASSWORDLESS AUTHENTICATION**

**Why needed:** Ansible requires passwordless SSH authentication to connect to target servers automatically.

**On shell-script-example server (Ansible control node):**

**Generate SSH key pair:**

ssh-keygen

**Press Enter for all prompts to accept defaults**

**Verify keys created:**

ls /home/ubuntu/.ssh/

**Expected output:**

id_rsa  id_rsa.pub  known_hosts

**Files explanation:**

- id_rsa: Private key (keep secret)
- id_rsa.pub: Public key (can be shared)

**View public key:**

cat /home/ubuntu/.ssh/id_rsa.pub

**Output example:**

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC... ubuntu@shell-script-example

**Copy this entire public key**

---

**STEP 3: ADD PUBLIC KEY TO TARGET SERVER**

**Login to test server (target server)**

**Generate SSH keys (if not already done):**

ssh-keygen

**Press Enter for all prompts**

**Check SSH directory:**

ls ~/.ssh

**Expected output:**

authorized_keys  id_rsa  id_rsa.pub

**Open authorized_keys file:**

vim ~/.ssh/authorized_keys

**Paste the public key from shell-script-example server**

**Save and exit:** Press Esc, then type :wq! and press Enter

---

**STEP 4: TEST PASSWORDLESS AUTHENTICATION**

**Get private IP of test server:** Example: 172.31.26.252

**From shell-script-example server, connect to test server:**

ssh 172.31.26.252

**Expected result:** You should connect WITHOUT password prompt

**Success output:**

Welcome to Ubuntu 22.04.1 LTS

ubuntu@ip-172-31-26-252:~$

**If it asks for password:** Passwordless authentication is not configured correctly. Repeat steps 2 and 3.

**Exit from test server:**

exit

---

**PART 2: ANSIBLE INVENTORY**

---

**STEP 5: CREATE INVENTORY FILE**

**What is inventory file:** List of servers that Ansible will manage

**On shell-script-example server, create inventory file:**

vim inventory

**Add IP address of target server:**

172.31.26.252

**For multiple servers:**

172.31.26.252

172.31.26.100

172.31.26.101

**Save and exit**

---

**STEP 6: CREATE INVENTORY WITH GROUPS**

**Why grouping:** Different servers have different purposes. Grouping allows running specific tasks on specific server types.

**Edit inventory file:**

vim inventory

**Add groups:**

[dbservers]

172.31.62.28

172.31.62.29

[webservers]

172.31.62.100

172.31.62.101

**Explanation:**

- [dbservers]: Group name for database servers
- [webservers]: Group name for web servers
- Add server IPs under each group

**Save and exit**

---

**PART 3: ANSIBLE AD-HOC COMMANDS**

---

**WHAT ARE AD-HOC COMMANDS**

**Definition:** One-line commands to perform quick tasks without writing a playbook

**When to use:**

- Quick one or two tasks
- Testing
- Simple operations

**When NOT to use:**

- Complex operations
- Multiple related tasks
- Operations you want to repeat

---

**STEP 7: BASIC AD-HOC COMMAND**

**Task: Create a file on target server**

**Command:**

ansible -i inventory all -m "shell" -a "touch devopsclass"

**Command breakdown:**

**ansible:** Ansible command

**-i inventory:** Use inventory file named "inventory"

**all:** Run on all servers in inventory

**-m "shell":** Use shell module

**-a "touch devopsclass":** Arguments - create file named devopsclass

**Expected output:**

172.31.26.252 | CHANGED | rc=0 >>

**Color coding:**

- Yellow/Orange: Task executed successfully, changes made
- Green: Task executed successfully, no changes needed

- Red: Task failed

**Verify on target server:**

**Login to test server and run:**

ls

**You should see:**

devopsclass

**File created successfully!**

---

**STEP 8: AD-HOC COMMANDS ON SPECIFIC GROUPS**

**Run on webservers only:**

ansible -i inventory webservers -m "shell" -a "touch webfile"

**Run on dbservers only:**

ansible -i inventory dbservers -m "shell" -a "touch dbfile"

**Run on all servers:**

ansible -i inventory all -m "shell" -a "touch allfile"

**Run on specific IP:**

ansible -i inventory 172.31.26.252 -m "shell" -a "touch specificfile"

---

**COMMON AD-HOC COMMAND EXAMPLES**

**1. Create directory:**

ansible -i inventory all -m "shell" -a "mkdir /tmp/mydir"

**2. Install package:**

ansible -i inventory all -m "apt" -a "name=nginx state=present" -b

**Note:** -b means become (use sudo)

**3. Copy file:**

ansible -i inventory all -m "copy" -a "src=/home/ubuntu/file.txt dest=/tmp/"

**4. Check disk space:**

ansible -i inventory all -m "shell" -a "df -h"

**5. Check memory:**

ansible -i inventory all -m "shell" -a "free -h"

**6. Restart service:**

ansible -i inventory all -m "service" -a "name=nginx state=restarted" -b

**7. Check service status:**

ansible -i inventory all -m "shell" -a "systemctl status nginx" -b

**8. Remove file:**

ansible -i inventory all -m "file" -a "path=/tmp/testfile state=absent"

**9. Create user:**

ansible -i inventory all -m "user" -a "name=john state=present" -b

**10. Run multiple commands:**

ansible -i inventory all -m "shell" -a "cd /tmp && ls -la"

---

**ANSIBLE MODULES REFERENCE**

**Common modules:**

**shell:** Run shell commands

**apt:** Manage apt packages (Ubuntu/Debian)

**yum:** Manage yum packages (CentOS/RHEL)

**copy:** Copy files to target

**file:** Manage files and directories

**service:** Manage services

**user:** Manage users

**command:** Run commands (simpler than shell)

**template:** Deploy Jinja2 templates

**git:** Manage git repositories

**Get help on any module:**

ansible-doc module_name

**Example:**

ansible-doc shell

ansible-doc apt

ansible-doc copy

**List all modules:**

ansible-doc -l

---

**PART 4: ANSIBLE PLAYBOOKS**

---

**WHAT IS AN ANSIBLE PLAYBOOK**

**Definition:** YAML file containing automation tasks

**When to use playbooks:**

- Multiple related tasks
- Complex operations
- Repeatable automation
- Configuration management

**Difference from ad-hoc commands:**

**Ad-hoc commands:**

- One or two tasks
- Command line
- Quick operations

**Playbooks:**

- Many tasks
- YAML file
- Complex workflows
- Reusable
- Version controlled

---

**STEP 9: CREATE FIRST PLAYBOOK**

**Task: Install and start nginx**

**Create playbook file:**

vim first-playbook.yml

**Playbook content:**

yaml

```
---
- name: Install and Start nginx
  hosts: all
  become: true
  become_user: root

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

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

**Save and exit**

---

**PLAYBOOK EXPLANATION LINE BY LINE**

**Line 1:**

yaml

```
---
```

YAML file start indicator (three dashes)

**Line 2:**

yaml

- name: Install and Start nginx

- Dash indicates start of play
- name: Descriptive name for this play

**Line 3:**

yaml

hosts: all

- Run on which servers
- Options: all, webservers, dbservers, specific IP

**Line 4:**

yaml

become: true

- Execute with elevated privileges (sudo)
- Required for installing packages and managing services

**Line 5:**

yaml

become_user: root

- Which user to become
- Typically root for system operations

**Line 7:**

yaml

tasks:

Start of tasks section

**Line 8-11: First task**

yaml

```
  - name: Install nginx
    apt:
```

name: nginx

state: present

- Task name: Install nginx
- Module: apt (package manager)
- Package name: nginx
- State: present (ensure installed)

**Line 13-16: Second task**

yaml

- name: Start nginx

service:

name: nginx

state: started

- Task name: Start nginx
- Module: service
- Service name: nginx
- State: started (ensure running)

---

**PLAYBOOK SYNTAX RULES**

**YAML syntax rules:**

**1. Indentation:**

- Use spaces, NOT tabs
- Consistent indentation (2 or 4 spaces)
- Children indented more than parents

**2. Lists:**

- Start with dash (-)
- Example:

yaml

tasks:

- name: Task 1

- name: Task 2

**3. Dictionaries:**

- Key: value pairs
- Example:

yaml

apt:

name: nginx

state: present

**4. Strings:**

- Can use quotes or not
- Use quotes for special characters

**5. Comments:**

- Start with #

yaml

*# This is a comment*

```

---

**STEP 10: RUN THE PLAYBOOK**

**Execute playbook:**

```

ansible-playbook -i inventory first-playbook.yml

```

**Command breakdown:**
```

**ansible-playbook:** Command to run playbooks

**-i inventory:** Specify inventory file

**first-playbook.yml:** Playbook filename

---

**UNDERSTANDING PLAYBOOK OUTPUT**

**Output example:**

```
PLAY [Install and Start nginx] ****************************************

TASK [Gathering Facts] ************************************************
ok: [172.31.26.252]

TASK [Install nginx] **************************************************
changed: [172.31.26.252]

TASK [Start nginx] ****************************************************
ok: [172.31.26.252]

PLAY RECAP ************************************************************
172.31.26.252   : ok=3   changed=1   unreachable=0   failed=0
```

**Output explanation:**

**PLAY [Install and Start nginx]:**

Play started

**TASK [Gathering Facts]:**

- First task always runs automatically

- Collects system information

- Verifies connectivity

- Checks passwordless authentication

**Status: ok**

- Green color

- Facts gathered successfully

**TASK [Install nginx]:**

Second task executing

**Status: changed**

- Yellow/Orange color

- Nginx was not installed before

- Now installed

- System state changed

**TASK [Start nginx]:**

Third task executing

**Status: ok**

- Green color

- Service started successfully

**PLAY RECAP:**

Summary of execution

**ok=3:** 3 tasks completed successfully

**changed=1:** 1 task made changes (installed nginx)

**unreachable=0:** All servers reachable

**failed=0:** No failures

**skipped=0:** No tasks skipped

**rescued=0:** No errors rescued

**ignored=0:** No errors ignored

---

**STATUS MEANINGS**

**ok (Green):**

- Task completed successfully

- No changes needed

- System already in desired state

**changed (Yellow/Orange):**

- Task completed successfully
- Changes were made
- System state modified

**failed (Red):**

- Task failed
- Error occurred
- System not in desired state

**skipped (Cyan):**

- Task skipped
- Conditional not met
- Not applicable

---

**STEP 11: VERIFY NGINX INSTALLATION**

**Login to target server**

**Check nginx status:**

```
sudo systemctl status nginx
```

**Expected output:**

```
● nginx.service - A high performance web server
   Loaded: loaded (/lib/systemd/system/nginx.service)
   Active: active (running) since Thu 2024-11-07 10:30:00 UTC
```

**Check if nginx is listening:**

```
sudo netstat -tlnp | grep nginx
```

**Access nginx:**

```
curl localhost
```

**Should return nginx welcome page HTML**

---

**STEP 12: RUN PLAYBOOK IN VERBOSE MODE**

**Debug mode (verbose):**

```
ansible-playbook -vvv -i inventory first-playbook.yml
```

**Verbosity levels:**

**-v:** Basic additional information

**-vv:** More detailed information

**-vvv:** Very detailed (shows execution steps)

**-vvvv:** Maximum verbosity (shows connection debugging)

**What verbose mode shows:**

- Detailed execution steps
- Module parameters
- Return values
- Connection details
- Timing information

**When to use:**

- Troubleshooting failures
- Understanding execution flow
- Debugging issues
- Learning how Ansible works

---

**PART 5: ADVANCED PLAYBOOK EXAMPLES**

---

**EXAMPLE 1: PLAYBOOK WITH VARIABLES**

**Create playbook:**

```
vim variables-playbook.yml
```

**Content:**

```
yaml
---
- name: Install package with variables
  hosts: webservers
  become: true
  vars:
    package_name: nginx
    service_name: nginx

  tasks:
    - name: Install package
      apt:
        name: "{{ package_name }}"
        state: present

    - name: Start service
      service:
        name: "{{ service_name }}"
        state: started
```

**Variables explanation:**

- vars: section defines variables

- Double curly braces {{ }} for variable usage

- Makes playbook reusable

---

**EXAMPLE 2: PLAYBOOK WITH LOOPS**

**Create playbook:**

```
vim loop-playbook.yml
```

**Content:**

```
yaml
---
- name: Install multiple packages
  hosts: all
  become: true

  tasks:
    - name: Install packages
      apt:
        name: "{{ item }}"
        state: present
      loop:
        - nginx
        - git
        - curl
        - wget
```

```

**Loop explanation:**

- loop: defines list of items

- {{ item }} represents current iteration

- Installs all packages in list

---

**EXAMPLE 3: PLAYBOOK WITH CONDITIONS**

**Create playbook:**

```
vim conditional-playbook.yml
```

**Content:**

yaml

```
---
- name: Conditional execution
  hosts: all
  become: true

  tasks:
    - name: Install nginx on Ubuntu
      apt:
        name: nginx
        state: present
      when: ansible_distribution == "Ubuntu"
```
```

```
  - name: Install nginx on CentOS
    yum:
      name: nginx
      state: present
    when: ansible_distribution == "CentOS"
```

**Condition explanation:**

- when: adds condition
- ansible_distribution: system fact
- Different tasks for different OS

---

**EXAMPLE 4: PLAYBOOK WITH HANDLERS**

**Create playbook:**

```
vim handler-playbook.yml
```

**Content:**

```
yaml
---
- name: Configure nginx
  hosts: webservers
  become: true

  tasks:
    - name: Install nginx
```

```
    apt:
      name: nginx
      state: present

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

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

**Handler explanation:**
- notify: triggers handler
- Handler runs only if task changes
- Handlers run at end of playbook
- Useful for service restarts

---

**EXAMPLE 5: MULTI-PLAY PLAYBOOK**

**Create playbook:**

```

vim multi-play-playbook.yml

**Content:**

yaml

---

- name: Configure database servers
  hosts: dbservers
  become: true
  tasks:
    - name: Install MySQL
      apt:
        name: mysql-server
        state: present

- name: Configure web servers
  hosts: webservers
  become: true
  tasks:
    - name: Install Apache
      apt:
        name: apache2
        state: present
```

**Multi-play explanation:**

- Multiple plays in one playbook
- Each play targets different hosts
- Different tasks for different server types

---

**PART 6: ANSIBLE ROLES**

---

**WHAT ARE ANSIBLE ROLES**

**Definition:**

Organized way to structure playbooks for reusability and maintainability

**Why use roles:**

- Large playbooks become unmanageable
- Hard to read
- Difficult to maintain
- Not reusable

**Benefits of roles:**

- Organized structure
- Reusable components
- Better readability
- Easier maintenance
- Team collaboration
- Version control friendly

---

**STEP 13: CREATE ANSIBLE ROLE**

**Create project directory:**

```
mkdir second-playbook
cd second-playbook
```

**Generate role structure:**

```
ansible-galaxy role init kubernetes
```

**Output:**

```
- Role kubernetes was created successfully
```

**Verify role created:**

```
ls
```

**Output:**

```
kubernetes
```

**Check role structure:**

```
cd kubernetes
ls
```

**Output:**

```
defaults files handlers meta README.md tasks templates tests vars
```

---

**ANSIBLE ROLE DIRECTORY STRUCTURE**

**Complete structure:**

```
kubernetes/
├── defaults/
│   └── main.yml
├── files/
├── handlers/
│   └── main.yml
├── meta/
│   └── main.yml
├── README.md
├── tasks/
│   └── main.yml
```

```
├── templates/
├── tests/
│   ├── inventory
│   └── test.yml
└── vars/
    └── main.yml
```

---

**ROLE FOLDERS EXPLANATION**

**1. tasks/ folder:**

**Purpose:** Contains main tasks

**File:** tasks/main.yml

**Description:**

- Core of the role
- All tasks to execute
- Main logic here

**Example tasks/main.yml:**

yaml

```
---
- name: Install Docker
  apt:
    name: docker.io
    state: present

- name: Install kubeadm
  apt:
    name: kubeadm
    state: present
```

- name: Start Docker

  service:

    name: docker

    state: started

---

**2. defaults/ folder:**

**Purpose:** Default variables

**File:** defaults/main.yml

**Description:**

- Lowest priority variables
- Can be easily overridden
- Default values for role

**Example defaults/main.yml:**

yaml

---

kubernetes_version: "1.28"

docker_version: "latest"

pod_network_cidr: "10.244.0.0/16"

**Priority:** Lowest (easily overridden)

---

**3. vars/ folder:**

**Purpose:** Role variables

**File:** vars/main.yml

**Description:**

- Higher priority than defaults
- Less likely to be overridden
- Role-specific values

**Example vars/main.yml:**

yaml

---

kubernetes_repo: "https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64"

docker_repo: "https://download.docker.com/linux/ubuntu"

**Priority:** Higher than defaults

---

**4. files/ folder:**

**Purpose:** Static files

**Files:** Any file type

**Description:**

- Store static files
- Copied to target as-is
- No variable substitution

**Examples:**

- Certificates
- Scripts
- Configuration files
- HTML files
- Images

**Usage in tasks:**

yaml

- name: Copy certificate

  copy:

    src: ssl-cert.crt

    dest: /etc/ssl/certs/

```
```

**Ansible looks for ssl-cert.crt in files/ folder**

---

**5. templates/ folder:**

**Purpose:** Jinja2 templates

**Files:** .j2 extension

**Description:**
- Dynamic content
- Variables replaced at runtime
- Uses Jinja2 templating engine

**Example template (config.j2):**
```
server_name: {{ hostname }}
port: {{ port_number }}
environment: {{ env_type }}
database_host: {{ db_host }}
```
**Example vars:**
yaml
```
hostname: webserver1
port_number: 8080
env_type: production
db_host: 172.31.10.20
```

**Result after templating:**

```

server_name: webserver1

port: 8080

environment: production

database_host: 172.31.10.20
```

**Usage in tasks:**

yaml

```
- name: Deploy config
  template:
    src: config.j2
    dest: /etc/app/config.conf
```

**Ansible:**

1. Reads config.j2 from templates/
2. Replaces variables
3. Copies result to target

---

**6. handlers/ folder:**

**Purpose:** Event handlers

**File:** handlers/main.yml

**Description:**

- Tasks triggered by notify
- Run at end of playbook
- Only if notified
- Only run once even if notified multiple times

**Example handlers/main.yml:**

yaml

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

- name: Reload nginx
  service:
    name: nginx
    state: reloaded

- name: Restart docker
  service:
    name: docker
    state: restarted
```

**Usage in tasks:**

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

**Flow:**

1. Config file copied
2. If changed, notify handler
3. Handler runs at end
4. Nginx restarted

---

**7. meta/ folder:**

**Purpose:** Role metadata

**File:** meta/main.yml

**Description:**

- Information about role
- Dependencies
- Platform support
- Author info

**Example meta/main.yml:**

yaml

```
---
galaxy_info:
 author: DevOps Team
 description: Kubernetes cluster setup
 company: Example Inc
 license: MIT
 min_ansible_version: 2.9
 platforms:
  - name: Ubuntu
    versions:
     - focal
     - jammy
 galaxy_tags:
  - kubernetes
  - docker
  - container

dependencies: []
```

**Dependencies example:**

yaml

dependencies:

 - role: common

 - role: docker

**When this role runs, common and docker roles run first**

---

**8. tests/ folder:**

**Purpose:** Role testing

**Files:**

- tests/inventory
- tests/test.yml

**Description:**

- Test role functionality
- Verify role works
- CI/CD integration

**Example test.yml:**

yaml

---

- hosts: localhost

 remote_user: root

 roles:

  - kubernetes

```

**Run tests:**

```

ansible-playbook tests/test.yml -i tests/inventory

---

**9. README.md file:**

**Purpose:** Documentation

**Contains:**

- Role description
- Requirements
- Variables
- Usage examples
- License

**Example README.md:**

markdown

# Kubernetes Role

This role installs and configures Kubernetes.

## Requirements

- Ubuntu 20.04 or later

- Ansible 2.9 or later

## Variables

- kubernetes_version: Version to install (default: 1.28)

- pod_network_cidr: Pod network CIDR (default: 10.244.0.0/16)

## Example Playbook

```yaml

- hosts: servers
```

```
  roles:
    - kubernetes
```

## License

MIT
```

---

**STEP 14: USE ROLES IN PLAYBOOK**

**Create main playbook:**
```
cd ..
vim site.yml
```
**Content:**
yaml
```
---
- hosts: all
  roles:
    - kubernetes
```

**Save and exit**

**Run playbook:**
```

```

ansible-playbook -i inventory site.yml

```

**What happens:**

1. Ansible looks for kubernetes folder

2. Reads kubernetes/tasks/main.yml

3. Executes all tasks

4. Uses files, templates, handlers as needed

5. Applies to all hosts

---

**STEP 15: ROLE WITH MULTIPLE PLAYS**

**Create playbook:**

```

vim complex-site.yml

**Content:**

yaml

---

- name: Setup database servers

  hosts: dbservers

  become: true

  roles:

    - common

    - database
```

```
- name: Setup web servers
  hosts: webservers
  become: true
  roles:
    - common
    - webserver
    - nginx
```

**Execution flow:**

1. Apply common and database roles to dbservers
2. Apply common, webserver, and nginx roles to webservers
3. Each role executes its tasks in order

---

**COMPLETE ROLE EXAMPLE**

**Scenario: Configure web server with nginx**

**Role structure:**

```
webserver/
├── defaults/
│   └── main.yml
├── files/
│   └── index.html
├── handlers/
```

```
│   └── main.yml
├── tasks/
│   └── main.yml
├── templates/
│   └── nginx.conf.j2
└── vars/
    └── main.yml
```

**defaults/main.yml:**

```yaml
---
nginx_port: 80
nginx_user: www-data
```

**vars/main.yml:**

```yaml
---
nginx_root: /var/www/html
```

**files/index.html:**

```html
<!DOCTYPE html>
<html>
<head>
  <title>Welcome</title>
</head>
<body>
  <h1>Welcome to Nginx!</h1>
</body>
</html>
```

**templates/nginx.conf.j2:**

```
server {
    listen {{ nginx_port }};
    server_name localhost;
    root {{ nginx_root }};
    index index.html;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

**tasks/main.yml:**

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

- name: Copy index page
  copy:
    src: index.html
    dest: "{{ nginx_root }}/index.html"

- name: Deploy nginx config
  template:
```

```
    src: nginx.conf.j2
    dest: /etc/nginx/sites-available/default
  notify: Restart nginx

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

**handlers/main.yml:**

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

**Main playbook (site.yml):**

```
yaml
---
- hosts: webservers
  become: true
  roles:
    - webserver
```

**Run:**

```
ansible-playbook -i inventory site.yml
```

---

**VARIABLE PRECEDENCE IN ROLES**

**Priority order (lowest to highest):**

1. defaults/main.yml (lowest)
2. vars/main.yml
3. Playbook vars
4. Playbook vars_files
5. Host vars
6. Group vars
7. Extra vars (-e flag) (highest)

**Example:**

**defaults/main.yml:**

yaml

port: 80

**vars/main.yml:**

yaml

port: 8080

**Playbook:**

yaml

- hosts: all
  roles:
    - webserver
  vars:
    port: 9090

```

**Command:**

```

ansible-playbook site.yml -e "port=3000"

```

**Result:** port = 3000 (extra vars wins)

---

**ANSIBLE GALAXY**

**What is Ansible Galaxy:**

- Repository for Ansible roles
- Community-contributed content
- Ready-to-use roles
- Share your roles

**Search for roles:**

```
ansible-galaxy search nginx
```

**Install role from Galaxy:**

```
ansible-galaxy install geerlingguy.nginx
```

**Installed location:**

```
~/.ansible/roles/geerlingguy.nginx
```

**Use in playbook:**
```

yaml

---

- hosts: webservers

  roles:

    - geerlingguy.nginx

```

**List installed roles:**

```

ansible-galaxy list

```

**Remove role:**

```

ansible-galaxy remove geerlingguy.nginx

```

**Install specific version:**

```

ansible-galaxy install geerlingguy.nginx,2.8.0

**Install from requirements file:**

**Create requirements.yml:**

yaml

---

- src: geerlingguy.nginx

  version: 2.8.0

- src: geerlingguy.mysql

  version: 3.3.0

```

**Install:**

```
ansible-galaxy install -r requirements.yml
```

---

**REAL-WORLD EXAMPLE: KUBERNETES CLUSTER**

**Scenario:**

Set up Kubernetes cluster with 1 master and 2 workers

**Approach:**

**Step 1: Use Terraform to create 3 EC2 instances**

**Step 2: Use Ansible to configure**

**Inventory file:**

```
[master]

172.31.10.10

[workers]

172.31.10.20

172.31.10.21
```
```

**Playbook:**

yaml

---

- name: Setup all nodes

  hosts: all

  become: true

  roles:

    - common

    - docker

    - kubernetes

- name: Initialize master

  hosts: master

  become: true

  roles:

    - kubernetes-master

- name: Join workers

  hosts: workers

  become: true

  roles:

    - kubernetes-worker

```

**Why separate playbooks:**

- Master needs different configuration

- Workers need join token from master

- Different roles for different purposes
```

---

**INTERVIEW QUESTIONS**

**Q1: What is Ansible?**

Configuration management tool using agentless architecture and push-based model to manage servers.

**Q2: What is passwordless authentication and why is it needed?**

SSH key-based authentication allowing Ansible to connect without password. Required for automation.

**Q3: What is inventory file?**

List of servers that Ansible manages. Can include groups for organizing servers.

**Q4: Difference between ad-hoc commands and playbooks?**

Ad-hoc: Quick one-off tasks from command line

Playbooks: Complex, reusable automation in YAML files

**Q5: What are Ansible roles?**

Organized structure for playbooks with directories for tasks, handlers, templates, files, etc. Makes playbooks reusable and maintainable.

**Q6: What is Gathering Facts task?**

Automatically runs first in every playbook. Collects system information and verifies connectivity.

**Q7: What are handlers?**

Special tasks triggered by notify. Run at end of playbook only if notified. Used for service restarts.

**Q8: What is Jinja2 templating?**

Template engine used by Ansible for dynamic content. Variables in templates replaced at runtime.

**Q9: How to run tasks on specific servers?**

Use groups in inventory file and specify group in playbook hosts parameter.

**Q10: What is the difference between files and templates folders?**

files/: Static files copied as-is

templates/: Dynamic files with variables replaced

---

**BEST PRACTICES**

**1. Use version control:**

Store playbooks and roles in Git

**2. Use roles for organization:**

Break large playbooks into roles

**3. Use variables:**

Make playbooks reusable

**4. Use handlers:**

Efficient service management

**5. Test before production:**

Test playbooks in dev environment

**6. Use verbose mode for debugging:**

-vvv flag for troubleshooting

**7. Document your playbooks:**

Add comments and README

**8. Use tags:**

Run specific parts of playbooks

**9. Use check mode:**

--check flag for dry run

**10. Keep secrets secure:**

Use Ansible Vault for passwords

---

**COMMON COMMANDS REFERENCE**

**Installation:**

```
sudo apt update
sudo apt install ansible
ansible --version
```

```

**SSH setup:**

```
ssh-keygen
cat ~/.ssh/id_rsa.pub
vim ~/.ssh/authorized_keys
```

**Ad-hoc commands:**

```
ansible -i inventory all -m "shell" -a "command"
ansible -i inventory group -m "module" -a "args" -b
```

**Playbook commands:**

```
ansible-playbook -i inventory playbook.yml
ansible-playbook playbook.yml -vvv
ansible-playbook playbook.yml --check
```

**Role commands:**

```
ansible-galaxy role init rolename
ansible-galaxy search keyword
ansible-galaxy install role
ansible-galaxy list
```
```

```

**Testing:**

```
ansible -i inventory all -m ping

ansible all -m setup
```

---

**TROUBLESHOOTING GUIDE**

**Issue 1: Connection refused**

**Check:**
- Target server running
- Correct IP address
- SSH service running on target

**Solution:**
```
ping target_ip

ssh target_ip

sudo systemctl status sshd
```

**Issue 2: Permission denied**
```

**Check:**

- Passwordless authentication configured

- Public key in authorized_keys

- Correct file permissions

**Solution:**

```
ssh-copy-id target_ip

chmod 600 ~/.ssh/id_rsa

chmod 644 ~/.ssh/id_rsa.pub
```

**Issue 3: Module not found**

**Check:**

- Correct module name

- Ansible version

**Solution:**

```
ansible-doc -l | grep module_name
```

**Issue 4: Playbook syntax error**

**Check:**

- YAML syntax

- Indentation (use spaces, not tabs)

**Solution:**

```
ansible-playbook playbook.yml --syntax-check
```

**Issue 5: Task failing**

**Check:**

- Run with verbose mode
- Check error message

**Solution:**

```
ansible-playbook playbook.yml -vvv
```