Ansible for DevOps Engineers Zero to Advanced with Real-World Examples

Author : *Umar Shahzad*

Introduction to Ansible

Ansible is an open-source automation tool used for configuration management, application deployment, and task automation. It is agentless, meaning it does not require software to be installed on remote systems. Ansible operates using SSH for Linux systems and WinRM for Windows systems.

Why Use Ansible?

- **Agentless Architecture**: No need to install agents on target machines.
- **Idempotency**: Ensures that running the same playbook multiple times does not cause unintended changes.
- **Declarative Language**: Uses YAML for configuration, making it easy to read and write.
- **Scalability**: Manages thousands of nodes efficiently.

Setting Up Ansible

Installation (Linux)

```
sudo apt update && sudo apt install ansible -y
```

OR (For RHEL-based systems)

```
sudo yum install ansible -y
```

Installation (MacOS)

```
brew install ansible
```

Verifying Installation

```
ansible --version
```

Configuring Ansible

Edit the inventory file (default location: `/etc/ansible/hosts`):

```
[webservers]
192.168.1.10
192.168.1.11

[dbservers]
192.168.1.20
```

Writing Your First Playbook

A playbook is a YAML file containing automation instructions.

Example: Installing Apache

```
- name: Install Apache Web Server
  hosts: webservers
  become: yes
  tasks:
    - name: Install Apache
      apt:
        name: apache2
```

```
        state: present
    - name: Start and Enable Apache
      service:
        name: apache2
        state: started
        enabled: yes
```

Run the playbook using:

```
ansible-playbook apache.yml
```

Advanced Ansible Features

Ansible Roles

Roles help structure playbooks by separating tasks, variables, and handlers.

```
ansible-galaxy init my_role
```

Directory structure:

```
my_role/
├── tasks/main.yml
├── handlers/main.yml
├── templates/
├── files/
├── vars/main.yml
├── defaults/main.yml
└── meta/main.yml
```

Ansible Vault (Encrypting Sensitive Data)

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

Ansible Loops (Running Tasks Multiple Times)

```
- name: Install multiple packages
  hosts: all
  tasks:
```

```
    - name: Install required packages
      apt:
        name: "{{ item }}"
        state: present
      loop:
        - vim
        - curl
        - git
```

Ansible Conditionals

```
- name: Install software based on OS
  hosts: all
  tasks:
    - name: Install Apache on Debian-based systems
      apt:
        name: apache2
        state: present
      when: ansible_os_family == "Debian"
```

Best Practices

1. **Use Roles for Scalability**: Keep your playbooks modular.
2. **Avoid Hardcoding Values**: Use variables and templates.
3. **Secure Sensitive Data**: Always use Ansible Vault.
4. **Use Dynamic Inventory**: Fetch inventory dynamically for cloud environments.
5. **Test in Staging Before Production**: Always validate changes in a non-production environment.
6. **Use Handlers for Efficient State Management**: Restart services only when required.
7. **Enable Logging for Troubleshooting**: Configure logs in `ansible.cfg`.

Use Cases

- **Cloud Automation**: Automate AWS, Azure, or GCP configurations.
- **CI/CD Pipelines**: Integrate with Jenkins for automated deployments.
- **Security Compliance**: Ensure systems comply with security policies.
- **Patch Management**: Automate OS and package updates across servers.
- **Application Deployment**: Deploy web applications seamlessly.

Expert Tips

- **Leverage `--check` Mode**: Test playbooks without making changes.
- **Optimize Performance**: Use `forks` and `async` tasks for speed.
- **Use Tags for Selective Execution**: Run specific parts of a playbook.

```
ansible-playbook playbook.yml --tags "install"
```

- **Monitor Playbook Execution**: Use Ansible Tower or AWX for visibility.
- **Use Debugging Tools**: Run tasks in verbose mode for better insights.

```
ansible-playbook playbook.yml -v
```

Enable Fact Caching: Speed up playbook execution by enabling fact caching.

```
[defaults]

fact_caching = jsonfile

fact_caching_connection = /tmp/ansible_facts
```

This Documents takes you from zero to advanced Ansible expertise with real-world examples, best practices, and expert insights. Master Ansible to automate infrastructure efficiently and enhance system reliability.

For more informative Documentations follow me on Linkedin .