Managing Configurations with Ansible

Comprehensive Guide to Configuration Management and Automation

Prepared By: Rashi Rana
Corporate Trainer

What is Configuration Management?

Configuration Management is the practice of systematically handling changes to a system in a way that maintains integrity over time. It involves automating the deployment, configuration, and management of infrastructure and applications.

Core Principles

- **Consistency:** Ensure all systems are configured identically across environments
- **Repeatability:** Ability to reproduce configurations reliably
- **Traceability:** Track all changes and maintain audit trails
- **Automation:** Reduce manual intervention and human errors

Key Benefits

Reduced Drift

Prevents configuration drift by maintaining desired state across all systems.

Faster Deployment

Automated provisioning and configuration significantly reduce deployment time.

Improved Reliability

Consistent configurations reduce system failures and unexpected behaviors.

Compliance

Ensures systems meet security and regulatory requirements consistently.

Traditional vs. Modern Approach

Traditional configuration management relied on manual processes, shell scripts, and documentation. Modern approaches use declarative tools that define desired state and automatically enforce it.

Prepared By: Rashi Rana
Corporate Trainer

Why Ansible?

Ansible Overview

Ansible is an open-source automation platform that simplifies configuration management, application deployment, and task automation through human-readable YAML syntax.

Key Advantages

Agentless Architecture

No need to install agents on managed nodes. Uses SSH for Linux/Unix and WinRM for Windows.

Simple Syntax

Uses YAML for playbooks, making it easy to read, write, and understand by both developers and operations teams.

Idempotent Operations

Running the same playbook multiple times produces the same result without side effects.

Extensive Module Library

Over 3,000+ modules for managing various systems, cloud platforms, and applications.

Ansible vs. Other Tools

Feature	Ansible	Puppet	Chef
Agent Required	No	Yes	Yes

Feature	Ansible	Puppet	Chef
Configuration Language	YAML	Puppet DSL	Ruby DSL
Learning Curve	Low	Medium	High
Push/Pull Model	Push	Pull	Pull
Setup Complexity	Simple	Complex	Complex

Prepared By: Rashi Rana
Corporate Trainer

YAML Basics

YAML (YAML Ain't Markup Language)

YAML is a human-readable data serialization standard used extensively in Ansible for writing playbooks, inventory files, and configuration files.

YAML Syntax Rules

- **Indentation:** Uses spaces (not tabs) for structure - typically 2 spaces
- **Case Sensitive:** Keys and values are case-sensitive
- **Key-Value Pairs:** Separated by colon and space (key: value)
- **Lists:** Items start with dash and space (- item)
- **Comments:** Start with hash symbol (#)

YAML Data Types

```
# Strings
name: "Apache Web Server"
description: 'Single quotes also work'
multiline: |
  This is a multiline
  string that preserves
  line breaks

# Numbers
port: 80
timeout: 30.5

# Booleans
enabled: true
debug: false
ssl_enabled: yes

# Lists
packages:
  - httpd
  - php
  - mysql

# Dictionaries
database:
  host: localhost
  port: 3306
  name: webapp
  user: dbuser
```

Common YAML Mistakes

- Using tabs instead of spaces for indentation
- Inconsistent indentation levels
- Missing space after colon in key-value pairs
- Incorrect list syntax (missing dash or space)

Prepared By: Rashi Rana
Corporate Trainer

Ansible Architecture

Architecture Overview

Control Node → Inventory → Playbooks → Managed Nodes

Core Components

Control Node

The machine where Ansible is installed and from which all tasks and playbooks are executed.

Managed Nodes

Target systems that are managed by Ansible. No agent installation required.

Inventory

List of managed nodes with connection details, groups, and variables.

Modules

Units of code that perform specific tasks like installing packages or managing services.

Playbooks

YAML files containing ordered lists of tasks to be executed on managed nodes.

Roles

Reusable collections of tasks, variables, files, and templates organized in a standard structure.

Execution Flow

1. **Read Inventory:** Ansible reads the inventory file to identify target hosts
2. **Parse Playbook:** Playbook is parsed and tasks are identified
3. **Establish Connection:** SSH connections are established to managed nodes
4. **Execute Modules:** Python modules are transferred and executed on target hosts
5. **Return Results:** Results are collected and reported back to the control node

Prepared By: Rashi Rana
Corporate Trainer

Ansible Inventory

What is Inventory?

Inventory is a file that defines the hosts and groups of hosts upon which commands, modules, and tasks in a playbook operate.

Inventory Formats

INI Format (Default)

```
# Static inventory file (hosts.ini)
[webservers]
web1.example.com
web2.example.com
192.168.1.10

[databases]
db1.example.com ansible_host=192.168.1.20
db2.example.com ansible_host=192.168.1.21

[production:children]
webservers
databases

[webservers:vars]
http_port=80
max_clients=200
```

YAML Format

```
all:
  children:
    webservers:
      hosts:
        web1.example.com:
        web2.example.com:
        192.168.1.10:
      vars:
        http_port: 80
        max_clients: 200
    databases:
      hosts:
        db1.example.com:
          ansible_host: 192.168.1.20
        db2.example.com:
          ansible_host: 192.168.1.21
```

Common Inventory Variables

Variable	Description	Example
ansible_host	IP address or hostname to connect to	ansible_host=192.168.1.10
ansible_port	SSH port number	ansible_port=2222
ansible_user	Username for SSH connection	ansible_user=ubuntu
ansible_ssh_private_key_file	Path to SSH private key	ansible_ssh_private_key_file=~/.ssh/id_rsa

Prepared By: Rashi Rana
Corporate Trainer

Ansible Playbooks

What are Playbooks?

Playbooks are YAML files that define a series of tasks to be executed on managed nodes. They are the foundation of Ansible's configuration management and deployment capabilities.

Playbook Structure

```
---
- name: Install and configure Apache
  hosts: webservers
  become: yes
  vars:
    http_port: 80
    document_root: /var/www/html
  tasks:
    - name: Install Apache package
      yum:
        name: httpd
        state: present

    - name: Start Apache service
      service:
        name: httpd
        state: started
        enabled: yes

  handlers:
    - name: restart apache
      service:
        name: httpd
        state: restarted
```

Key Playbook Components

Plays

Tasks

Top-level structure that maps hosts to tasks. Each play targets specific hosts and defines tasks to execute.

Individual units of work that call Ansible modules to perform specific actions on managed nodes.

Variables

Dynamic values that can be used throughout playbooks to make them flexible and reusable.

Handlers

Special tasks that run only when notified by other tasks, typically used for service restarts.

Playbook Execution

```
# Run a playbook
ansible-playbook -i inventory.ini playbook.yml

# Run with specific tags
ansible-playbook -i inventory.ini playbook.yml --tags "install"

# Dry run (check mode)
ansible-playbook -i inventory.ini playbook.yml --check

# Verbose output
ansible-playbook -i inventory.ini playbook.yml -vvv
```

Prepared By: Rashi Rana
Corporate Trainer

Ansible Modules

What are Modules?

Modules are discrete units of code that perform specific tasks. Ansible ships with over 3,000 modules that can manage various aspects of systems, applications, and cloud resources.

Common Module Categories

System Modules

user, group, cron, service
Manage system users, groups, scheduled tasks, and services.

Package Modules

yum, apt, pip, npm
Install, update, and remove packages using various package managers.

File Modules

copy, template, file, lineinfile
Manage files, directories, and file content on managed nodes.

Cloud Modules

ec2, azure_rm, gcp_compute
Manage cloud resources across AWS, Azure, GCP, and other providers.

Popular Modules Examples

```
# Package management
- name: Install packages
  yum:
    name: ['httpd', 'php', 'mysql']
    state: present
```

```yaml
# File operations
- name: Copy configuration file
  copy:
    src: /local/path/httpd.conf
    dest: /etc/httpd/conf/httpd.conf
    backup: yes
  notify: restart apache

# Service management
- name: Ensure Apache is running
  service:
    name: httpd
    state: started
    enabled: yes

# Command execution
- name: Run custom script
  command: /usr/local/bin/setup.sh
  args:
    creates: /var/log/setup.log
```

Module Documentation

```bash
# Get module documentation
ansible-doc yum

# List all modules
ansible-doc -l

# Search for modules
ansible-doc -l | grep -i mysql
```

Prepared By: Rashi Rana
Corporate Trainer

Ansible Roles

What are Roles?

Roles are a way to organize playbooks and reuse code. They provide a standardized directory structure for organizing tasks, variables, files, templates, and handlers.

Role Directory Structure

```
roles/
└── apache/
    ├── tasks/
    │   └── main.yml
    ├── handlers/
    │   └── main.yml
    ├── templates/
    │   └── httpd.conf.j2
    ├── files/
    │   └── index.html
    ├── vars/
    │   └── main.yml
    ├── defaults/
    │   └── main.yml
    ├── meta/
    │   └── main.yml
    └── README.md
```

Role Components

Directory	Purpose	File Type
tasks/	Main list of tasks to be executed	YAML
handlers/	Handlers triggered by tasks	YAML

Directory	Purpose	File Type
templates/	Jinja2 templates for dynamic files	.j2
files/	Static files to be copied	Any
vars/	Role-specific variables	YAML
defaults/	Default variables (lowest priority)	YAML
meta/	Role metadata and dependencies	YAML

Using Roles in Playbooks

```
---
- name: Configure web servers
  hosts: webservers
  become: yes
  roles:
    - apache
    - php
    - mysql
```

Creating Roles

```
# Create role structure
ansible-galaxy init apache

# Install role from Ansible Galaxy
ansible-galaxy install geerlingguy.apache

# List installed roles
ansible-galaxy list
```

Prepared By: Rashi Rana
Corporate Trainer

Lab: Writing Playbooks - Tasks, Variables, Handlers

Understanding Playbook Components

1. Tasks

Tasks are the basic unit of work in Ansible. Each task calls a module with specific parameters.

```
tasks:
  - name: Install Apache web server
    yum:
      name: httpd
      state: present
    tags: install

  - name: Create document root
    file:
      path: "{{ document_root }}"
      state: directory
      mode: '0755'
```

2. Variables

Variables make playbooks flexible and reusable across different environments.

```
vars:
  http_port: 80
  https_port: 443
  document_root: /var/www/html
  server_name: "{{ ansible_hostname }}"
  packages:
    - httpd
    - mod_ssl
    - php
```

3. Handlers

Handlers are tasks that run only when notified by other tasks, typically for service restarts.

```
handlers:
  - name: restart apache
    service:
      name: httpd
      state: restarted

  - name: reload apache
    service:
      name: httpd
      state: reloaded
```

Prepared By: Rashi Rana
Corporate Trainer

Hands-on: Install Ansible

Prerequisites

- **Control Node:** Linux/macOS system with Python 3.8+
- **Managed Nodes:** EC2 instances with SSH access
- **Network:** SSH connectivity between control and managed nodes
- **Privileges:** sudo access on managed nodes

Installation Methods

Method 1: Using pip (Recommended)

```
# Update system packages
sudo apt update && sudo apt upgrade -y

# Install Python and pip
sudo apt install python3 python3-pip -y

# Install Ansible
pip3 install ansible

# Verify installation
ansible --version
ansible-playbook --version
```

Method 2: Using Package Manager (Ubuntu/Debian)

```
# Add Ansible PPA
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update
ppa:ansible/ansible

# Install Ansible
```

```
sudo apt install ansible

# Verify installation
ansible --version
```

Method 3: Using Package Manager (RHEL/CentOS)

```
# Enable EPEL repository
sudo yum install epel-release -y

# Install Ansible
sudo yum install ansible -y

# Verify installation
ansible --version
```

Prepared By: Rashi Rana
Corporate Trainer

EC2 Instance Setup for Labs

AWS EC2 Instance Configuration

1 **Launch EC2 Instances:**

- 2 Control Node: 1 instance (t2.micro or larger)
- 3 Managed Nodes: 2-3 instances (t2.micro)
- 4 AMI: Ubuntu 20.04 LTS or Amazon Linux 2
- 5 Key Pair: Create or use existing SSH key pair

6 **Security Group Configuration:**

```
# Inbound Rules
SSH (22) - Source: Your IP or 0.0.0.0/0
HTTP (80) - Source: 0.0.0.0/0
HTTPS (443) - Source: 0.0.0.0/0

# Outbound Rules
All Traffic - Destination: 0.0.0.0/0
```

7 **SSH Key Setup:**

```
# Copy private key to control node
scp -i your-key.pem your-key.pem ubuntu@control-
node-ip:~/.ssh/

# Set proper permissions
chmod 600 ~/.ssh/your-key.pem

# Test SSH connectivity
ssh -i ~/.ssh/your-key.pem ubuntu@managed-node-ip
```

8 **Configure SSH Agent (Optional):**

```
# Start SSH agent
eval $(ssh-agent)

# Add private key
ssh-add ~/.ssh/your-key.pem

# Test passwordless SSH
ssh ubuntu@managed-node-ip
```

Prepared By: Rashi Rana
Corporate Trainer

Sample Playbook: Install Apache on EC2

Step 1: Create Inventory File

```
# Create inventory.ini
[webservers]
web1 ansible_host=3.15.24.156 ansible_user=ubuntu
web2 ansible_host=18.191.171.45 ansible_user=ubuntu

[webservers:vars]
ansible_ssh_private_key_file=~/.ssh/your-key.pem
ansible_ssh_common_args='-o StrictHostKeyChecking=no'
```

Step 2: Create Apache Installation Playbook

```
---
- name: Install and Configure Apache Web Server
  hosts: webservers
  become: yes
  vars:
    http_port: 80
    document_root: /var/www/html
    server_admin: admin@example.com

  tasks:
    - name: Update package cache (Ubuntu)
      apt:
        update_cache: yes
      when: ansible_os_family == "Debian"

    - name: Install Apache (Ubuntu)
      apt:
        name: apache2
        state: present
      when: ansible_os_family == "Debian"

    - name: Install Apache (Amazon Linux)
```

```
        yum:
          name: httpd
          state: present
        when: ansible_os_family == "RedHat"
```

Prepared By: Rashi Rana
Corporate Trainer

Complete Apache Playbook

apache-install.yml (Continued)

```
    - name: Create custom index.html
      copy:
        content: |
          <html>
          <head><title>Welcome to {{ ansible_hostname }}
</title></head>
          <body>
            <h1>Hello from {{ ansible_hostname }}</h1>
            <p>Server IP: {{ ansible_default_ipv4.address
}}</p>
            <p>Managed by Ansible</p>
          </body>
          </html>
        dest: "{{ document_root }}/index.html"
        mode: '0644'

    - name: Start and enable Apache (Ubuntu)
      service:
        name: apache2
        state: started
        enabled: yes
      when: ansible_os_family == "Debian"

    - name: Start and enable Apache (Amazon Linux)
      service:
        name: httpd
        state: started
        enabled: yes
      when: ansible_os_family == "RedHat"

    - name: Open firewall for HTTP (Amazon Linux)
      firewalld:
        service: http
        permanent: yes
        state: enabled
        immediate: yes
        when: ansible_os_family == "RedHat"
        ignore_errors: yes
```

Step 3: Execute the Playbook

```
# Test connectivity
ansible -i inventory.ini webservers -m ping

# Run the playbook
ansible-playbook -i inventory.ini apache-install.yml

# Verify Apache installation
ansible -i inventory.ini webservers -m shell -a
"systemctl status apache2"

# Test web server
curl http://<server-ip>
```

Prepared By: Rashi Rana
Corporate Trainer

Practical Exercises

Exercise 1: Basic Ansible Commands

1. **Test Connectivity:**

```
# Ping all hosts
ansible -i inventory.ini all -m ping

# Check disk space
ansible -i inventory.ini webservers -m shell -a
"df -h"

# Get system information
ansible -i inventory.ini webservers -m setup
```

2. **Ad-hoc Commands:**

```
# Install package
ansible -i inventory.ini webservers -m apt -a
"name=htop state=present" --become

# Create user
ansible -i inventory.ini webservers -m user -a
"name=testuser state=present" --become

# Copy file
ansible -i inventory.ini webservers -m copy -a
"src=/tmp/test.txt dest=/tmp/test.txt" --become
```

3. **Gather Facts:**

```
# Collect system facts
ansible -i inventory.ini webservers -m setup |
grep ansible_distribution
```

```
# Filter specific facts
ansible -i inventory.ini webservers -m setup -a
"filter=ansible_memory_mb"
```

Exercise 2: Advanced Playbook Features

- Add conditional tasks based on OS family
- Use loops to install multiple packages
- Implement error handling with ignore_errors
- Add tags for selective execution
- Use templates for configuration files

Prepared By: Rashi Rana
Corporate Trainer

Best Practices and Troubleshooting

Ansible Best Practices

Playbook Organization

Use roles for complex configurations, maintain clear directory structure, and version control all code.

Variable Management

Use group_vars and host_vars directories, encrypt sensitive data with Ansible Vault.

Idempotency

Ensure tasks can be run multiple times safely, use appropriate modules and conditions.

Error Handling

Use failed_when, ignore_errors, and rescue blocks for robust error handling.

Common Issues and Solutions

Issue	Cause	Solution
SSH Connection Failed	Wrong key, user, or host	Verify SSH connectivity manually first
Permission Denied	Missing sudo privileges	Use --become flag or become: yes in playbook
Module Not Found	Typo in module name	Check module documentation with ansible-doc

Issue	Cause	Solution
YAML Syntax Error	Indentation or formatting issues	Use YAML validator or ansible-playbook --syntax-check

Debugging Commands

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

# Dry run
ansible-playbook playbook.yml --check

# Verbose output
ansible-playbook playbook.yml -vvv

# Step through tasks
ansible-playbook playbook.yml --step
```

Prepared By: Rashi Rana
Corporate Trainer

Summary and Next Steps

Key Concepts Covered

- Configuration Management principles and benefits
- Ansible's agentless architecture and key advantages
- YAML syntax fundamentals for writing playbooks
- Ansible architecture: Control nodes, managed nodes, inventory, and modules
- Playbook structure: tasks, variables, and handlers
- Practical experience with EC2 instances and Apache installation

Practical Skills Acquired

- Installing and configuring Ansible on control nodes
- Setting up EC2 instances for Ansible management
- Writing inventory files with host groups and variables
- Creating and executing playbooks for software installation
- Using ad-hoc commands for quick system management tasks
- Troubleshooting common Ansible issues and errors

Recommended Next Steps

1. **Advanced Playbooks:** Explore conditionals, loops, and error handling
2. **Ansible Roles:** Create reusable roles for complex configurations
3. **Ansible Vault:** Learn to encrypt sensitive data and passwords
4. **Dynamic Inventory:** Integrate with cloud providers for automatic host discovery

5. **Ansible Tower/AWX:** Explore enterprise features and web-based management
6. **CI/CD Integration:** Incorporate Ansible into deployment pipelines

Additional Resources

- Ansible Official Documentation: https://docs.ansible.com/
- Ansible Galaxy: https://galaxy.ansible.com/ (Community roles and collections)
- Ansible GitHub Repository: https://github.com/ansible/ansible
- Red Hat Ansible Automation Platform: Enterprise solutions and support

Prepared By: Rashi Rana
Corporate Trainer