# Introduction to Ansible

## Understand with Real Time Implementation Example




**DigiTalk**

**https://digitalksystems.com/**

**Reach us at digitalk.fmw@gmail.com**
**DigiTalk Channel: https://www.youtube.com/channel/UCCGTnI9vvF_ETMhGUXGdFWw**
**Playlists: https://www.youtube.com/@digitalk.middleware/playlists**
**Weblogic Server Architecture: https://youtu.be/gNqeIfLjUqw**

# DigiTalk Udemy Courses and Coupon Code

**SOA Suite Administration**
https://www.udemy.com/course/mastering-oracle-soa-suite-12c-administration/?couponCode=739A60915F86847014EB
Coupon Code: 739A60915F86847014EB

**JBoss 8 Administration**
https://www.udemy.com/course/mastering-jboss-eap-8-administration-from-intro-to-advanced/?couponCode=BF65EB008CFE16686BD2
Coupon Code:BF65EB008CFE16686BD2

**OHS Administration**
https://www.udemy.com/course/mastering-oracle-ohs-http-12c-web-server-administration/?couponCode=8E990556B21AF3E1A316
Coupon Code: 8E990556B21AF3E1A316

**Weblogic Server Administration**
https://www.udemy.com/course/oracle-weblogic-server-12c-and-14c-administration/?couponCode=87BC1314AC7690FD5294
Coupon Code:87BC1314AC7690FD5294

You can write us on digitalk.fmw@gmail.com if coupon code expired.

## Ansible in Real Life: The Dinner Party Analogy

**Scenario**: Imagine you're hosting a large dinner party at your home. You need to prepare and serve the same meal to all your guests, and you want everything to be perfectly organized.

### 1. Planning the Menu

- **Without Ansible**: You would need to personally cook each dish, set the table, and ensure everything is served hot and on time. This would be a lot of work, especially if you have many guests. You might forget some details or mess up the timing.
- **With Ansible**: You can create a detailed plan (like a recipe) for the meal preparation and serving process. This plan includes all the steps you need to follow to cook the meal, set the table, and serve it.

### 2. Using a Recipe Card (Playbook)

- **Recipe Card**: You write down the recipe and the steps needed to prepare the meal. This includes what ingredients to use, how to cook each dish, how to set the table, and how to serve the meal.

Example Recipe Card:

  - Cook spaghetti.
  - Prepare tomato sauce.
  - Set the table with plates, forks, and glasses.
  - Serve spaghetti and sauce to each guest.

### 3. Assembling a Team (Inventory)

- **Assembling a Team**: You gather a team of friends or family members to help with different tasks. Each person follows the steps on the recipe card but does their part (e.g., cooking spaghetti, preparing the sauce, setting the table).

**Example Team:**

  - Alice cooks the spaghetti.
  - Bob prepares the sauce.
  - Carol sets the table.
  - Dave serves the meal.

### 4. Executing the Plan

- **Without Ansible**: You'd have to check on each task manually, make sure each person knows what to do, and handle any issues that arise.
- **With Ansible**: You hand over the recipe card to your team. They follow the instructions to the letter, ensuring each task is done consistently. If any issues come up (e.g., running out of an ingredient), you have a clear plan to handle it, and everyone is aware of what needs to be done.

### 5. Benefits in Real Life

- **Consistency**: Every guest receives the same meal, prepared in the same way, because everyone followed the same recipe.
- **Efficiency**: The meal preparation is faster and more organized because each person knows exactly what to do.
- **Scalability**: If you decide to host another party or need to serve more guests, you can use the same recipe card and team to replicate the process.

### Summary

In this analogy, Ansible is like a well-organized recipe card that you use to ensure a dinner party goes smoothly. It provides a clear set of instructions (playbook) to your team (servers) and helps ensure everything is prepared and served consistently. Just as the recipe card helps manage the dinner party efficiently, Ansible automates tasks and configurations in IT, ensuring consistent and efficient results.

## Introduction to Ansible

Ansible is an open-source automation tool used for configuration management, application deployment, and task automation. It enables IT teams to automate repetitive tasks, manage infrastructure, and orchestrate complex workflows in a simple and efficient manner.

### Key Components and Concepts of Ansible

#### 1. Control Node:

**Definition:** The machine where Ansible is installed and from which it runs tasks. It orchestrates the automation tasks and communicates with managed nodes.

**Example:** If you are using Ansible to manage a fleet of servers, your local machine or a dedicated server with Ansible installed acts as the control node.

#### 2. Managed Nodes:

**Definition:** The servers or devices that Ansible manages and configures. These nodes receive and execute the automation tasks sent from the control node.

**Example:** Web servers, database servers, and network devices that Ansible configures or deploys applications on. If you have a web server that needs to be updated, it would be a managed node.

#### 3. Inventory:

**Definition:** A file or script that lists all the managed nodes. It defines the hosts and their groups, allowing Ansible to know where to apply the configurations.

**Example:** An inventory file (inventory.ini) might list all your servers with their IP addresses or hostnames, organized into groups such as [webservers] and [databases].

#### 4. Playbooks:

**Definition:** YAML files that define a set of tasks to be executed on managed nodes. Playbooks describe the desired state of the system and the actions to achieve it.

**Example:** A playbook to install and configure Apache on a web server might include tasks for updating packages, installing Apache, and starting the service.

**5. Roles:**

**Definition:** A way to organize playbooks into reusable components. Roles group tasks, variables, files, and templates into a structured directory.

**Example:** A role named apache might include tasks to install and configure Apache, along with related files and templates. This role can be reused across multiple playbooks.

**6. Modules:**

**Definition:** Units of work in Ansible that perform specific tasks, such as installing software or managing files. Modules are the building blocks of playbooks.

**Example:** The yum module installs packages on Red Hat-based systems, while the copy module copies files to managed nodes.

**7. Tasks:**

**Definition:** Individual actions defined in a playbook or role that are executed on managed nodes. Tasks use modules to perform their work.

**Example:** A task might use the service module to start the Apache service on a server.

**8. Handlers:**

**Definition:** Special tasks that are triggered by other tasks when changes occur. Handlers are used for actions like restarting a service after a configuration change.

**Example:** A handler might restart the Apache service if a configuration file is modified.

**9. Variables:**

**Definition:** Values that can be used to parameterize playbooks and roles. Variables allow you to customize configurations and make playbooks more flexible.

**Example:** A variable might define the version of a software package to be installed, allowing the same playbook to work with different versions.

**10. Templates:**

**Definition:** Files that are processed by the Jinja2 templating engine to generate dynamic content. Templates are used to create configuration files with variable data.

**Example:** A template might generate a custom Apache configuration file based on variables defined in the playbook.

## Real-Life Example

**Scenario: Automating the deployment of a web application on multiple servers.**

**Control Node Setup:**

Install Ansible on a local machine or a dedicated server. This will be the control node where you execute Ansible commands.

**Inventory Configuration:**

Create an inventory file (**inventory.ini**) listing your web servers:

```
[webservers]

web1.example.com

web2.example.com
```

**Playbook Creation:**

Write a playbook (**deploy-web.yml**) to install and configure Apache:

```
- hosts: webservers
  become: yes
  tasks:
    - name: Update package index
      apt:
        update_cache: yes
    - name: Install Apache
      apt:
        name: apache2
        state: present
    - name: Start Apache
      service:
        name: apache2
        state: started
        enabled: yes
```

**Role Development:**

Create a role directory (roles/apache) with tasks to install and configure Apache, using reusable components.

**Running Ansible:**

Execute the playbook from the control node:

ansible-playbook -i inventory.ini deploy-web.yml

**Verification:**

Ansible connects to each web server (managed node), installs Apache, starts the service, and ensures it is enabled on boot.

## Summary

Ansible is a powerful automation tool for managing and configuring IT infrastructure. It simplifies automation through a centralized control node, with managed nodes receiving and executing tasks. Key components include playbooks for defining tasks, roles for organizing components, and modules for performing specific actions. In a real-life scenario, Ansible automates tasks such as deploying a web application across multiple servers, streamlining operations and ensuring consistency.

## Understand with Few More Real Life Scenario

## Deploying a Web Application

**Scenario: Imagine you are managing the deployment of a web application to multiple servers in a data center.**

### Real-Life Example 1:

**Control Node:** Your development workstation or a dedicated server where Ansible is installed.

**Managed Nodes:** The web servers where the application will be deployed.

**Inventory:** A list of web servers:

```
[webservers]
web1.example.com
web2.example.com
web3.example.com
```

**Playbook:** Automate tasks such as installing necessary packages, deploying code, and configuring web servers:

```
- hosts: webservers
  become: yes
  tasks:
    - name: Update package index
      apt:
        update_cache: yes
    - name: Install Nginx
      apt:
        name: nginx
        state: present
    - name: Deploy application code
      copy:
        src: /local/path/to/app/
        dest: /var/www/html/
```

```
    - name: Start Nginx
  service:
    name: nginx
    state: started
    enabled: yes
```

Execution: Run the playbook to deploy the application:

ansible-playbook -i inventory.ini deploy-web.yml

**Outcome:** Ansible connects to each web server, updates the package index, installs Nginx, deploys the application code, and starts the Nginx service.

**Benefits in the Real World**

**Consistency:** Ensures that all web servers are configured in the same way, reducing the risk of discrepancies and errors.

**Efficiency:** Automates repetitive tasks, saving time and effort compared to manual configuration.

**Scalability:** Easily scale up by adding more servers to the inventory file and running the same playbook without additional manual steps.

**Reproducibility:** If you need to redeploy or update the application, you can simply run the playbook again, ensuring that the deployment is consistent each time.

### Real-Life Example 2:

## Configuring a Database Cluster

**Scenario: You need to configure and deploy a MySQL database cluster across several servers.**

**Real-Life Example:**

**Control Node:** Your local machine with Ansible installed.

**Managed Nodes:** Database servers that need to be configured.

**Inventory:** A list of database servers:

```
[databases]
db1.example.com
db2.example.com
db3.example.com
```

**Playbook:** Automate the installation and configuration of MySQL:

```
- hosts: databases
  become: yes
  tasks:
    - name: Install MySQL server
      apt:
```

```
      name: mysql-server
      state: present
  - name: Ensure MySQL service is running
    service:
      name: mysql
      state: started
      enabled: yes
  - name: Configure MySQL replication
    template:
      src: /local/path/to/my.cnf.j2
      dest: /etc/mysql/my.cnf
    notify:
      - Restart MySQL
handlers:
  - name: Restart MySQL
    service:
      name: mysql
      state: restarted
```

**Execution:** Run the playbook to configure the database cluster:

ansible-playbook -i inventory.ini configure-database.yml

**Outcome:** Ansible installs MySQL on each database server, configures replication, and ensures the service is running and enabled.

### Real-Life Example 3:

## Setting Up a CI/CD Pipeline

**Scenario: You are setting up a continuous integration and deployment (CI/CD) pipeline for a development team.**

### Real-Life Example:

**Control Node:** A server running Jenkins, where Ansible is used to automate its setup.

**Managed Nodes:** Servers where Jenkins and its dependencies will be installed.

**Inventory:** A list of servers:

```
[ci_cd]
jenkins.example.com
```

**Playbook:** Automate the installation and configuration of Jenkins:

```
- hosts: ci_cd
  become: yes
  tasks:
    - name: Add Jenkins repository
```

```
    apt_repository:
      repo: 'deb http://pkg.jenkins.io/debian-stable binary/'
      state: present
  - name: Install Jenkins
    apt:
      name: jenkins
      state: present
  - name: Ensure Jenkins is started
    service:
      name: jenkins
      state: started
      enabled: yes
```

**Execution:** Run the playbook to set up Jenkins:

ansible-playbook -i inventory.ini setup-jenkins.yml

Outcome: Ansible installs Jenkins on the server, ensuring it is running and configured for use.

### Real-Life Example 3:

## Network Device Configuration

**Scenario: Configuring network switches and routers in a data center.**

**Real-Life Example:**

**Control Node:** Your network management system where Ansible is used.

**Managed Nodes:** Network devices such as switches and routers.

**Inventory:** A list of network devices:

```
[network]
switch1.example.com
switch2.example.com
router1.example.com
```

Playbook: Automate network device configuration:

```
- hosts: network
  tasks:
    - name: Configure VLANs
      ios_config:
        lines:
          - vlan 10
          - name Accounting
          - vlan 20
          - name Engineering
    - name: Set interface IP addresses
      ios_config:
```

```
    lines:
      - interface GigabitEthernet0/1
      - ip address 192.168.1.1 255.255.255.0
```

**Execution:** Run the playbook to configure network devices:

```
ansible-playbook -i inventory.ini configure-network.yml
```

**Outcome:** Ansible configures VLANs and IP addresses on the network devices as specified.

## Summary

Ansible is a versatile automation tool used across various scenarios, including deploying web applications, configuring databases, setting up CI/CD pipelines, and managing network devices. By defining tasks in playbooks and organizing them into roles and inventories, Ansible simplifies and automates complex IT operations, ensuring consistency and efficiency.

## DISCLAIMER AND CONSENT

This document is being provided by DigiTalk as part of its effort to assist users in understanding and working with Ansible. While every effort has been made to ensure the accuracy and reliability of the information presented in this document, there is a possibility of typographical errors or inaccuracies. DigiTalk does not guarantee the correctness or completeness of the content provided in this document.

Users of this document are encouraged to cross-reference the information presented here with official documentation available on their website or other authoritative sources. Any discrepancies or inaccuracies found in this document should be reported to us at digitalk.fmw@gmail.com.

By using this document, you acknowledge and consent to the following:

This document is not officially endorsed or verified by any other third party organization..

The Company makes no claims or guarantees about the accuracy or suitability of the information contained in this document.

Users are responsible for verifying and validating any information presented here for their specific use case.

DigiTalk disclaims any liability for any errors, omissions, or damages that may result from the use of this document.

If you discover any inaccuracies or errors in this document, please report them to digitalk.fmw@gmail.com, and the Company will endeavor to correct them as necessary.

This consent statement is provided to ensure transparency and understanding of the limitations of the information contained in this document. By using this document, you agree to abide by the terms and conditions outlined herein.