**Introduction to Ansible - Basic Study Material**

**What is Ansible?**

Ansible is an open-source automation tool used for configuration management, application deployment, and task automation. It allows users to automate IT infrastructure with simple, human-readable scripts written in YAML.

**Core Components of Ansible**

**1. Inventory**

The inventory file is a list of managed hosts stored in a file. This file can be in **INI** or **YAML** format. By default, Ansible uses the inventory located at /etc/ansible/hosts.

**Example of Inventory File (INI format)**

[webservers]

web1.example.com

web2.example.com

[dbservers]

db1.example.com

db2.example.com

In the above example, there are two groups: webservers and dbservers, each containing multiple hosts.

**2. Modules**

Modules are scripts that perform specific tasks like installing packages, copying files, or restarting services. Each module accepts parameters and returns output in JSON format.

**Example of a Module Usage**

The following command installs nginx on a remote machine:

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

**3. Variables**

Variables help in managing system differences and making playbooks dynamic. Variables can be stored in **dictionaries** or **lists**.

**Example of Variables in Playbooks**

vars:

database\_name: mydb

destination: /etc/config/

Variables can also be grouped based on host or playbook level.

**4. Facts**

Facts are system information automatically gathered by Ansible. These include OS type, IP addresses, and memory usage.

**Example of Fact Gathering**

To display facts about a host, run:

ansible all -m setup

**5. Playbooks**

Playbooks define automation tasks in **YAML format**. They contain multiple plays that map groups of hosts to specific tasks.

**Example of a Simple Playbook**

- name: Install and start Apache

hosts: webservers

become: yes

tasks:

- name: Install Apache

apt:

name: apache2

state: present

- name: Start Apache

service:

name: apache2

state: started

**6. Configuration File**

Ansible uses a configuration file (ansible.cfg) to override default settings. The order of configuration file lookup is:

1. ansible.cfg in the current directory
2. ~/.ansible.cfg in the home directory
3. /etc/ansible/ansible.cfg (default)

**Example of an Ansible Configuration File**

[defaults]

inventory = ./inventory

host\_key\_checking = False

retry\_files\_enabled = False

**7. Ad Hoc Commands**

Ad hoc commands are used for executing quick tasks without writing a playbook. They are useful for one-time tasks like checking logs, managing services, or verifying package installations.

**Example of an Ad Hoc Command**

The following command checks system information on localhost:

ansible localhost -m setup

**Difference Between Ad Hoc Commands and Playbooks**

| **Feature** | **Ad Hoc Command** | **Playbook** |
| --- | --- | --- |
| Execution | Single command | YAML script |
| Use Case | One-time task | Large deployments |
| Syntax | Command-line | Structured YAML |

**Commonly Used Modules in Ad Hoc Commands**

| **Module** | **Purpose** |
| --- | --- |
| ping | Checks if a server is reachable |
| setup | Gathers system facts |
| apt | Manages packages on Ubuntu/Debian |
| yum | Manages packages on RHEL/CentOS |
| service | Manages system services |
| user | Adds or removes users |
| copy | Copies files to remote systems |

**Summary Table**

| **Component** | **Description** |
| --- | --- |
| **Inventory** | List of managed hosts (INI/YAML format) |
| **Modules** | Predefined tasks executed by Ansible |
| **Variables** | Store data dynamically for tasks |
| **Facts** | Automatically gathered system information |
| **Playbooks** | YAML-based automation scripts |
| **Configuration File** | Controls Ansible's default behavior |
| **Ad Hoc Commands** | One-time tasks executed via command-line |

**Conclusion**

Ansible simplifies automation using YAML-based playbooks and eliminates the need for manual configuration. Understanding these basic components will help in efficiently managing and automating IT tasks.

**Ansible Ad Hoc Commands - Beginner's Guide**

**What is an Ansible Ad Hoc Command?**

Ansible Ad Hoc commands allow you to quickly execute a single task on a remote system without writing a full playbook. These commands are typically used for one-time tasks, such as installing a package, restarting a service, or gathering system information.

**When to Use Ad Hoc Commands?**

* When you need to execute a quick task without writing a playbook.
* Checking system logs or configurations.
* Installing or removing software packages.
* Restarting or stopping services.

**Syntax of Ansible Ad Hoc Commands**

An Ansible Ad Hoc command follows this syntax:

ansible <host-group> -i <inventory-file> -m <module> -a <arguments> [-b]

Where:

* <host-group>: Specifies the target host(s) from the inventory file.
* -i <inventory-file>: Specifies the inventory file containing the list of managed hosts.
* -m <module>: Specifies the module to execute (e.g., apt, yum, service).
* -a <arguments>: Specifies the module parameters (e.g., package name, service name).
* -b (optional): Runs the command with elevated privileges (sudo/root access).

**Example: Installing a Package**

Let's try to install the vim package using the apt module on a local machine.

ansible localhost -m apt -a "name=vim state=latest" -b

**Explanation:**

* localhost: Specifies the target machine.
* -m apt: Uses the apt module (package manager for Debian-based systems).
* -a "name=vim state=latest": Installs the latest version of vim.
* -b: Runs the command as a privileged user (root).

**Expected Output:**

If the package is installed successfully, Ansible will return a success message. If vim is already installed, Ansible will skip the step and show no changes.

**Example: Removing a Package**

If we want to remove vim, we change the state to absent:

ansible localhost -m apt -a "name=vim state=absent" -b

**Expected Output:**

Ansible will confirm that the package has been removed.

**Handling Permissions with -b (Become)**

If you try to install or remove a package without root privileges, Ansible will return an error. To fix this, we use the -b flag to execute the command as root.

ansible localhost -m apt -a "name=vim state=latest" -b

This allows Ansible to perform administrative tasks without switching users manually.

**Summary Table**

| **Command** | **Description** |
| --- | --- |
| ansible localhost -m apt -a "name=vim state=latest" -b | Installs the vim package |
| ansible localhost -m apt -a "name=vim state=absent" -b | Removes the vim package |
| ansible localhost -m service -a "name=nginx state=started" -b | Starts the nginx service |
| ansible localhost -m ping | Checks if the host is reachable |

By using Ad Hoc commands, you can quickly perform system administration tasks without writing long playbooks. For more complex automation, consider using Ansible Playbooks.

**Ansible Ad-Hoc Commands - Study Material**

**What is an Ansible Ad-Hoc Command?**

An Ansible ad-hoc command is a simple, one-time command used to perform quick administrative tasks on remote servers. Unlike playbooks, which are reusable scripts, ad-hoc commands are used for immediate execution of tasks without writing a full script.

**When to Use Ad-Hoc Commands?**

* Checking system logs
* Restarting services
* Installing or removing software
* Gathering system information
* Managing files and users

**Syntax of an Ansible Ad-Hoc Command**

ansible <host\_group> -i <inventory\_file> -m <module> -a "<module\_arguments>" [-b]

**Explanation of Syntax**

* <host\_group>: The target server(s) where the command will run.
* -i <inventory\_file>: Specifies the inventory file containing server details.
* -m <module>: Defines the module to use (e.g., apt, file, ping).
* -a "<module\_arguments>": Provides arguments to the module.
* -b: Runs the command with sudo privileges if required.

**Example Commands**

**1. Checking if a Server is Reachable**

ansible all -m ping

**Output:**

{"ping": "pong"}

**2. Installing a Package (Example: vim)**

ansible localhost -m apt -a "name=vim state=latest" -b

This installs the latest version of vim on the target system.

**3. Removing a Package**

ansible localhost -m apt -a "name=vim state=absent" -b

This removes vim from the system.

**4. Creating a File**

ansible localhost -m file -a "path=./example.txt state=touch"

Creates an empty file example.txt in the current directory.

**5. Deleting a File**

ansible localhost -m file -a "path=./example.txt state=absent"

Removes the example.txt file.

**Difference Between Ad-Hoc Commands and Playbooks**

| **Feature** | **Ad-Hoc Command** | **Playbook** |
| --- | --- | --- |
| Execution | One-time task | Reusable script |
| Flexibility | Simple and quick | More structured and automated |
| Best for | Quick tasks | Large-scale automation |

**File Permissions Example**

When creating a file, you can set permissions using the mode parameter:

ansible localhost -m file -a "path=./secure.txt state=touch mode=0444"

This creates a file with read-only permissions.

**Summary**

* Ansible ad-hoc commands are useful for quick tasks.
* They use modules like apt, file, ping, etc.
* Commands can install, remove packages, manage files, and check system status.
* They are different from playbooks, which are reusable scripts.

This material provides a fundamental understanding of Ansible ad-hoc commands with examples. 🚀