Ansible Inventory - Beginner's Guide

What is Ansible Inventory?

Ansible Inventory is a file that keeps a list of all the servers (also called hosts) that Ansible manages. It tells Ansible where to run commands and playbooks.

Purpose of Inventory

- It helps in managing multiple servers efficiently.
- Defines groups of servers based on their roles.
- Stores server-specific or group-specific variables.
- Supports static (fixed) and dynamic (changing) inventories.

Types of Inventory

1. Static Inventory

- A manually created file with a fixed list of servers.
- Stored in INI or YAML format.
- Requires manual updates when new servers are added or removed.

Example (INI Format)

```
[webservers]
server1.example.com
server2.example.com

[dbservers]
server3.example.com
```

Example (YAML Format)

```
all:
  hosts:
    server1.example.com:
    server2.example.com:
  children:
    webservers:
      hosts:
        server1.example.com:
        server2.example.com:
```

```
dbservers:
 hosts:
  server3.example.com:
```

2. Dynamic Inventory

- Generated automatically by querying external sources like AWS, Azure, or databases.
- Useful for cloud environments where IPs change frequently.
- Requires a script (Python, Bash, etc.) to fetch the latest server list.

Example (Dynamic Inventory using AWS EC2)

```
{
 "webservers": {
  "hosts": ["ec2-34-123-45-67.compute-1.amazonaws.com"],
  "vars": {
   "ansible_ssh_user": "ubuntu"
  }
 }
}
```

Difference Between Static and Dynamic Inventory

Feature	**Static Inventory**	**Dynamic Inventory**
Format	INI, YAML	JSON, Script-based
Updates	Manual	Automatic
Best for	Small infrastructures	Large, dynamic environments
Example Use Case	Fixed servers in a data center	Cloud-based servers with changing IPs

Inventory File Location

- Default location: /etc/ansible/hosts
- Custom location can be specified using -i flag:
- ansible -i my_inventory.ini all -m ping

Grouping Servers in Inventory

- Servers can be grouped based on roles like web servers, database servers, etc.

Example

```
[webservers]
server1.example.com
```

server2.example.com

[dbservers]

server3.example.com

[all_servers:children]

webservers

dbservers

Using Variables in Inventory

- Variables store extra information like SSH user, ports, or custom settings.

Example

[webservers]

server1.example.com ansible_ssh_user=ubuntu ansible_port=22

Best Practices

1. **Use Groups Wisely:** Organize servers into groups for easier management.
2. **Store Variables Separately:** Keep variables in group_vars/ and host_vars/ instead of the inventory file.
3. **Use Dynamic Inventory for Cloud Environments:** Helps in automatic server management.

Conclusion

Ansible Inventory is a critical part of Ansible automation. It helps in managing multiple servers effectively, whether they are static or dynamic. Understanding its structure and best practices ensures smooth infrastructure management with Ansible.

Dynamic Inventory in Ansible

What is Dynamic Inventory?

Ansible's **Dynamic Inventory** is a way to automatically fetch real-time infrastructure details rather than manually maintaining a static inventory file. It allows Ansible to gather host information dynamically from various sources such as AWS, Azure, Google Cloud, OpenStack, and more.

Why Use Dynamic Inventory?

- Automates inventory management, reducing manual updates.
- Useful for cloud environments where hosts frequently change (e.g., auto-scaling instances in AWS).
- Ensures accuracy in host details at runtime.

How Dynamic Inventory Works

Unlike **Static Inventory** (which is manually defined in a file), **Dynamic Inventory** is generated using an **executable script** or **plugin** that fetches hosts from an external system (like AWS, OpenStack, etc.).

Key Characteristics:

- **Executable Scripts:** Must return host information in JSON format.
- **Works in Real-time:** Always fetches updated inventory when the playbook runs.
- **Supports Various Formats:** Scripts can be written in Python, Ruby, Bash, Go, etc.
- **Supports Filters:** Can fetch details for all hosts or specific ones.

Example of Dynamic Inventory Script

A dynamic inventory script should follow Ansible's expected JSON format. Below is a simple Python example:

```
#!/usr/bin/env python3

import json


def get_inventory():
    inventory = {
        "web_servers": {
            "hosts": ["192.168.1.10", "192.168.1.11"],
            "vars": {"ansible_user": "ubuntu"}
        },
        "db_servers": {
            "hosts": ["192.168.1.20"],
            "vars": {"ansible_user": "admin"}
        },
        "_meta": {"hostvars": {}}
    }
    print(json.dumps(inventory))


if __name__ == "__main__":
    get_inventory()
```

Steps to Use This Script:

1. Save the script as dynamic_inventory.py.
2. Make it executable:
3. chmod +x dynamic_inventory.py
4. Run the script to test output:
5. ./dynamic_inventory.py

This should return JSON output with host details.

6. Use it with Ansible:
7. ansible -i dynamic_inventory.py all --list-hosts

Dynamic Inventory with Cloud Providers

Ansible provides **pre-built dynamic inventory plugins** for cloud providers like AWS, Azure, and Google Cloud.

Cloud Provider	Plugin Name
AWS	amazon.aws.ec2
Azure	azure.azcollection.azure_rm
Google Cloud	google.cloud.gcp_compute
OpenStack	openstack.cloud.openstack

Example: AWS Dynamic Inventory

To use AWS dynamic inventory:

1. Install the required collection:
2. ansible-galaxy collection install amazon.aws
3. Configure AWS credentials (~/.aws/credentials).
4. Use AWS inventory plugin:
5. ansible-inventory -i aws_ec2.yml --list

AWS Inventory File Example (aws_ec2.yml):

```
plugin: amazon.aws.ec2
regions:
  - us-east-1
keyed_groups:
  - key: tags.Name
    prefix: instance
```

Running Playbooks with Dynamic Inventory

Once the dynamic inventory script or plugin is set up, you can use it in playbooks:

ansible-playbook -i dynamic_inventory.py site.yml

Or for AWS:

ansible-playbook -i aws_ec2.yml site.yml

Summary

- **Dynamic Inventory** allows Ansible to fetch real-time host details from external sources.
- It is useful for **cloud environments** where servers frequently change.
- Scripts should return **JSON-formatted output** for Ansible.
- Ansible provides **pre-built plugins** for cloud providers like AWS, Azure, and GCP.

Using Dynamic Inventory makes infrastructure management **efficient, scalable, and automated**. 🚀

Using Inventory Variables in Ansible

What are Inventory Variables?

In Ansible, **inventory variables** are used to store information related to hosts or groups of hosts. These variables help manage configurations dynamically without modifying playbooks directly.

Why Use Inventory Variables?

- To store reusable data for hosts/groups.
- To simplify playbooks by avoiding hardcoded values.
- To make automation flexible and maintainable.

Inventory Structure

Example Inventory File (INI Format)

[tomcat]

localhost

Group Variables File (group_vars/tomcat.yml)

tomcat_port: 8080

Host Variables File (host_vars/localhost.yml)

host_name: localhost

Here:

- The **inventory file** defines a group called tomcat with one host (localhost).
- The **group variables file** (group_vars/tomcat.yml) defines tomcat_port as 8080.
- The **host variables file** (host_vars/localhost.yml) defines host_name as localhost.

Running Commands with Variables

You can use Ansible ad-hoc commands to retrieve variables.

Check a Variable's Value:

ansible tomcat -i inventory -m debug -a "msg={{ tomcat_port }}"

Output:

localhost | SUCCESS => {

"msg": "8080"

}

This confirms that the tomcat_port variable is correctly fetched from the group_vars file.

Creating a File Using Variables

We can create a file and write variable values into it using Ansible's file module.

Command to Create a File:

ansible localhost -i inventory -m copy -a "content='Port: {{ tomcat_port }}' dest=/tmp/tomcat_config.txt"

Verify File Content:

cat /tmp/tomcat_config.txt

Output:

Port: 8080

Modifying a File Attribute Using Variables

To modify a file, we can use Ansible's file module.

Example Command:

ansible localhost -i inventory -m file -a "path=/tmp/tomcat_config.txt owner=root mode=0644"

This changes the file ownership and permissions dynamically.

Conclusion

- Inventory variables help manage configurations dynamically.
- Variables can be defined at **host** or **group** level.
- Ansible ad-hoc commands can be used to retrieve and apply variables.
- Playbooks and modules like copy and file make use of inventory variables for automation.

By structuring inventory properly and using variables effectively, you can simplify your Ansible automation setup!

Ansible Dynamic Inventory Using a Script

Introduction

Ansible uses an **inventory** to track the hosts it manages. A static inventory contains a fixed list of hosts, while a **dynamic inventory** retrieves the list of hosts dynamically from an external source, such as a cloud provider (AWS, GCP, Azure, etc.). This allows Ansible to manage hosts that are automatically added or removed.

One common way to create a dynamic inventory is by using a script. In this guide, we will discuss how to set up and use a script for dynamic inventory in Ansible.

What is a Dynamic Inventory Script?

A dynamic inventory script is an executable file that generates an inventory dynamically. The script must output JSON format, which Ansible can understand.

Why Use a Dynamic Inventory?

- Automatically discovers hosts without manually updating the inventory file.
- Useful for cloud environments where servers are frequently created and destroyed.
- Reduces human errors and simplifies infrastructure management.

Setting Up a Dynamic Inventory Script

Step 1: Creating the Script

We will create a Python script named dynamic.py to fetch AWS EC2 instances as an inventory.

Example: A Basic Dynamic Inventory Script

```
#!/usr/bin/env python3

import json


inventory = {
   "all": {
      "hosts": ["ec2-34-201-120-25.compute-1.amazonaws.com"],
```

```
        "vars": {
            "ansible_user": "ubuntu",
            "ansible_ssh_private_key_file": "~/.ssh/aws-key.pem"
        }
    }
}

print(json.dumps(inventory))
```

Step 2: Making the Script Executable

To run the script, it must be executable. Use the following command:

```
chmod +x dynamic.py
```

Step 3: Running the Script Manually

Test the script by running:

```
./dynamic.py
```

Expected Output:

```
{
    "all": {
        "hosts": ["ec2-34-201-120-25.compute-1.amazonaws.com"],
        "vars": {
            "ansible_user": "ubuntu",
            "ansible_ssh_private_key_file": "~/.ssh/aws-key.pem"
        }
    }
}
```

Using the Dynamic Inventory in Ansible

Step 4: Running an Ansible Command Using the Dynamic Inventory

Now, we can use our script as an inventory source for Ansible:

```
ansible all -i dynamic.py -m ping
```

Step 5: Running a Playbook with a Dynamic Inventory

To run a playbook using the dynamic inventory:

```
ansible-playbook -i dynamic.py playbook.yml
```

Diagram: Static vs. Dynamic Inventory

Feature	Static Inventory	Dynamic Inventory
Definition	Fixed list of hosts	Hosts fetched dynamically
Maintenance	Manual updates needed	Automatically updates
Use Case	Small, fixed servers	Cloud-based, frequently changing servers

Conclusion

A dynamic inventory script is an essential tool for managing cloud-based infrastructure with Ansible. It automatically discovers and updates the list of hosts, reducing manual work and ensuring that the inventory remains up to date.

By using an executable script like dynamic.py, we can integrate external sources such as AWS, GCP, or databases to fetch hosts dynamically and use them in Ansible automation.

Now, you can implement dynamic inventories in your Ansible projects and manage hosts more efficiently! 🚀