Notes

Ansible Part 2

Divya Satpute

Learn Ansible Automation: A Beginner to Expert Guide

Welcome to the Ansible Adventure! 🌍✨

Hello, automation enthusiasts! 🎉 Ready to embark on a thrilling journey through the world of Ansible? Whether you're just starting or looking to supercharge your skills, this guide will walk you through everything from the basics to advanced techniques. Grab your gear, and let's dive into the magical world of automation! 🧑‍🚀🛠️

1. Getting Started: Your First Steps with Ansible 🏁🔧

- **What's Ansible?** 🤔: Ansible is a powerful automation tool that uses simple YAML files to define and execute tasks across your infrastructure. It's like having a personal assistant who follows your exact instructions to set up servers, deploy applications, and more!
- 📦 **Ansible setup** 📦

  Install packages

```
$ sudo apt-add-repository ppa:ansible/ansible

$sudo apt update -y

$sudo apt install ansible -y

$ansible --version
```

```
ubuntu@ip-172-31-39-94:~$ ansible --version
ansible [core 2.16.9]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/ubuntu/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules
']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/ubuntu/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Apr 10 2024, 05:33:47) [GCC 13.2.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
```

Learn Ansible Automation: A Beginner to Expert Guide

- ## Configuration Management 🌟

Configuration Management: Automate server setups, software installations, and system configurations.

- ## Push Based vs Pull Based

Tools like Puppet and chef are pull based

Agent on the server periodically checks for the configuration information from central server (master)

Ansible is Push Based

Central server pushes the configuration information on target servers. You control when the changes are made on the servers

(Ansible send notification to host servers to perform task)

- ## Inventory File 🔧

Ansible's inventory hosts file is used to list and group your servers. its default location is /etc/ansible/hosts

Note: In inventory host file we can mention IP address or Hostname also

Inventory 📋: This is your list of servers. It's where you define which servers Ansible should manage. For example:

```
[webservers]

server1.example.com

server2.example.com
```

Learn Ansible Automation: A Beginner to Expert Guide

Ansible Playbooks

🌟 What is an Ansible Playbook?

An Ansible playbook is a YAML file that defines a set of tasks to be executed on remote systems. It's a way to automate tasks like software installation, configuration changes, and system management. Think of it as a recipe for your infrastructure! 📜🍽️

- Playbooks 📜: Think of playbooks as detailed recipes. They outline the steps needed to achieve your goals. Here's a basic example:

```
- name: Install and start Apache
  hosts: webservers
  tasks:
    - name: Install Apache
      apt:
        name: apache2
        state: present
    - name: Start Apache
      service:
        name: apache2
        state: started
```

This playbook tells Ansible to install Apache on your web servers and then start the service.

Learn Ansible Automation: A Beginner to Expert Guide

2. Adding Spice: Intermediate Techniques 🌶️🚀

Now that you've got the basics, let's add some flavor:

Vault in Ansible 🔐

📚 What is Ansible Vault?

Ansible Vault is a feature of Ansible that allows you to keep sensitive data such as passwords or keys in encrypted files, rather than as plaintext in your playbooks or roles. This ensures that sensitive information is protected even if your playbook is shared or stored in a public repository.

🔑 **Key Features**

- **Encryption and Decryption**: Securely encrypt and decrypt files.
- **Ease of Use**: Simple commands to integrate encryption into your workflow.

Granular Control: Encrypt entire files or specific variables.

🛠️ How to Use Ansible Vault

1. Creating Encrypted Files

To create a new encrypted file, use the ansible-vault create command:

```
$ansible-vault create secrets.yml
```

This command will open your default text editor, allowing you to enter the sensitive data. Once you save and exit the editor, the file will be encrypted.

2. Encrypting Existing Files

You can encrypt an existing file using the ansible-vault encrypt command:

```
$ansible-vault encrypt existing-file.yml
```

3. Editing Encrypted Files

To edit an encrypted file, use the ansible-vault edit command:

```
$ansible-vault edit secrets.yml
```

This command will decrypt the file, open it in your default text editor, and then re-encrypt it once you save and exit.

4. Decrypting Files

If you need to decrypt a file, use the ansible-vault decrypt command:

$ansible-vault decrypt secrets.yml

5. Viewing Encrypted Files

To view the contents of an encrypted file without editing it, use the ansible-vault view command:

$ansible-vault view secrets.yml

🌟 Understanding Handlers in Ansible 🌟

📚 What are Handlers?

Handlers are special tasks in Ansible that are triggered only when notified by other tasks. They are typically used to perform actions that should only occur if there has been a change in the system state. Common use cases include restarting a service after a configuration file has been updated or reloading a web server when its configuration changes.

🔑 Key Characteristics of Handlers

- Conditional Execution: Handlers are only run when notified by another task.
- Idempotency: They ensure actions are only performed when necessary.
- Organization: They help keep playbooks clean and manageable.
- 🛠️ How Handlers Work

1. Define a Handler

Handlers are defined in the same way as regular tasks but are placed under the `handlers` section.

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

2. Notify a Handler

To notify a handler, use the notify directive in the task that may require the handler to run.

```
tasks:
  - name: copy apache configuration file
    copy:
      src: /path/to/httpd.conf
      dest: /etc/httpd/conf/httpd.conf
    notify:
      - restart apache
```

3. Handler Execution

Handlers are executed at the end of the playbook run, ensuring that the tasks have made all necessary changes before the handler is triggered.

🌀 Exploring Ansible Galaxy 🌀

📚 What is Ansible Galaxy?

Ansible Galaxy is a free service provided by Ansible that allows users to share and download roles. Roles in Ansible are a way of bundling automation content and can include tasks, handlers, variables, templates, and more. Galaxy helps you avoid reinventing the wheel by leveraging the work of the community.

🌟 Key Features

- **Discoverability**: Easily find and use roles created by the community.
- **Reusability**: Share your roles with others and reuse roles created by others.
- **Community Contributions**: Benefit from the expertise and contributions of a large community.

🔍 Finding Roles on Ansible Galaxy

You can browse roles on the Ansible Galaxy website or use the `ansible-galaxy` command-line tool to search for roles.

🔗 Using the Ansible Galaxy Website

Visit Ansible Galaxy to browse roles by category, popularity, or name. You can also see detailed information about each role, including documentation, dependencies, and ratings.

🔧 Using the `ansible-galaxy` Command

To search for roles from the command line, use the ansible-galaxy search command:

$ansible-galaxy search <role_name>

For example, to search for Nginx roles:

$ansible-galaxy search nginx

📦 Installing Roles

Once you've found a role you want to use, you can install it using the ansible-galaxy install command.

📝 Example Installation

To install a role named geerlingguy.nginx:

```
$ansible-galaxy install geerlingguy.nginx
```

The role will be downloaded and stored in your default roles path (/etc/ansible/roles or roles/ in your project directory).

🔧 Using Roles in Playbooks

After installing a role, you can include it in your playbooks. Here's an example of how to use the geerlingguy.nginx role in a playbook:

```
---
- hosts: webservers
  roles:
    - geerlingguy.nginx
```

🛠 Creating and Sharing Roles

Learn Ansible Automation: A Beginner to Expert Guide

Ansible Roles 🛠️

Roles are abstraction on top of tasks and playbooks that let you structure your Ansible configuration in a modular and reusable format.

As you add more and more functionality and flexibility to your playbooks, they can become unwieldy and difficult to maintain.

Roles allow you to break down a complex playbook into separate , smaller chunks that can be coordinated by a central entry point.

Key Components of a Role:

- Tasks 📝: The actions that the role performs.
- Handlers 🚦: Special tasks that are triggered by other tasks (e.g., restarting a service after configuration changes).
- Variables 🔑: Settings that customize the role's behavior.
- Templates 🖌️: Jinja2 templates used to create dynamic configuration files.
- Files 📂: Static files that the role needs (e.g., configuration files).
- Defaults ⚙️: Default values for variables that can be overridden.

Creating a Role: Your First Step to Organization 🚀🏗️

Let's walk through creating a role from scratch. Imagine we're building a role to set up a web server. Here's how you can do it:

1. **Create the Role Directory** 📂:

```
$ansible-galaxy init webserver
```

This command creates a directory named `webserver` with a predefined structure:

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

2. **Define Tasks** 🛠️:

In webserver/tasks/main.yml, write the tasks to install and configure your web server:

```
- name: Install Apache
  apt:
    name: apache2
    state: present
- name: Start Apache
  service:
    name: apache2
    state: started
```

3. **Add Handlers** 🚦:

In webserver/handlers/main.yml, define any handlers that should be triggered:

```
- name: Restart Apache
  service:
    name: apache2
    state: restarted
```

4. **Set Default Variables** 🔑:

In webserver/defaults/main.yml, set default values for variables:

```
apache_port: 80
```

5. Create Templates 📎:

Place your Jinja2 templates in webserver/templates/. For instance, apache.conf.j2 might look like this:

```
<VirtualHost *:{{ apache_port }}>
    DocumentRoot /var/www/html
</VirtualHost>
```

6. **Define Role Metadata** 📋:

```
galaxy_info:
  author: Your Name
  description: Role for setting up Apache web server
  company: Your Company
```

7. **Using Roles in Playbooks: Putting It All Together** 🧩📜

Once you have your role ready, using it in a playbook is straightforward. Here's how you can apply the webserver role:

```
- name: Set up web servers

  hosts: webservers

  roles:

    - webserver
```

This playbook applies the webserver role to all hosts in the webservers group. Ansible will automatically look in the webserver role directory for the tasks, handlers, variables, and templates.

Congratulations! You've unlocked the secrets of Ansible roles and are ready to organize and reuse your automation tasks like a pro.

3. Advanced Techniques: Unlocking the Power! 🔒💪

Advanced Techniques: Unlocking the Power! 🔒💪

🌍 Managing Ansible Inventories for Different Environments 🌍

In any robust infrastructure, managing different environments (such as development, testing, staging, and production) is essential. Ansible provides flexible inventory management, allowing you to define and use different inventories for various environments. This post will guide you through setting up and managing Ansible inventories for different environments efficiently.

📁 What is an Ansible Inventory?

An Ansible inventory is a collection of hosts (or nodes) that Ansible manages. It can be a simple static file, a dynamic script, or a combination of both. By organizing your inventory, you can target specific groups of hosts for different environments and streamline your automation processes.

📋 Static Inventory

Static inventories are the simplest way to define your hosts and groups. You can create separate inventory files for each environment.

📝 Example Inventory Files

Development Inventory (inventories/dev)

```
[webservers]

dev-web1 ansible_host=192.168.1.10

[dbservers]

dev-db1 ansible_host=192.168.1.11
```

Staging Inventory (inventories/staging)

```
[webservers]

staging-web1 ansible_host=192.168.2.10

[dbservers]

staging-db1 ansible_host=192.168.2.11
```

Production Inventory (inventories/prod)

```
[webservers]

prod-web1 ansible_host=192.168.3.10

[dbservers]

prod-db1 ansible_host=192.168.3.11
```

📦 Using Static Inventory Files

You can specify the inventory file when running Ansible commands:

$ansible-playbook -i inventories/dev playbook.yml

$ansible-playbook -i inventories/staging playbook.yml

$ansible-playbook -i inventories/prod playbook.yml

🌐 Dynamic Inventory

Dynamic inventories allow you to generate inventory data dynamically from external sources such as cloud providers, databases, or custom scripts.

📋 Example: AWS EC2 Dynamic Inventory

1. Install the boto3 library:

   $pip install boto3

2. Create a dynamic inventory script **(**aws_ec2.py**)**:

```
#!/usr/bin/env python

import boto3
import json

def get_instances():
    ec2 = boto3.client('ec2')
    instances = ec2.describe_instances()

    inventory = {'_meta': {'hostvars': {}}}
    for reservation in instances['Reservations']:
        for instance in reservation['Instances']:
            if instance['State']['Name'] == 'running':
                instance_id = instance['InstanceId']
                inventory['_meta']['hostvars'][instance_id] =
instance

    return inventory

if __name__ == '__main__':
    print(json.dumps(get_instances()))
```

3. Use the dynamic inventory script:

   $ansible-playbook -i aws_ec2.py playbook.yml

Learn Ansible Automation: A Beginner to Expert Guide

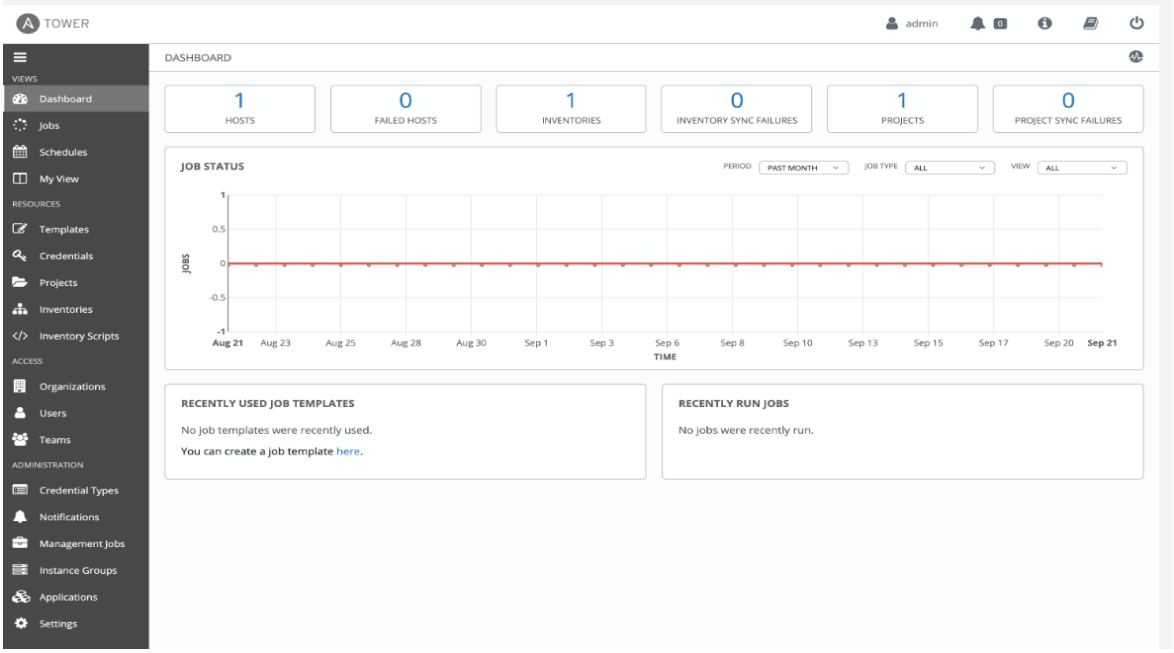
Automation with Ansible Tower 🏰

What is Ansible Tower? 🏰

Ansible Tower is a robust framework built on top of Ansible that provides a user-friendly web interface, role-based access control, job scheduling, and integrated notifications. It transforms your Ansible playbooks into a centralized automation tool that's easy to manage and scale.

Key Features:

- Web-based UI: Manage your Ansible playbooks, inventory, and jobs through an intuitive interface.
- Real-time Job Status: Monitor the progress of your automation tasks in real-time.
- Role-Based Access Control: Secure your environment by controlling who can run and manage jobs.
- Job Scheduling: Automate tasks to run at specific times or intervals.
- Integrated Notifications: Get alerts on job status via email, Slack, or other channels.
- REST API: Integrate Ansible Tower with other tools and systems.



Divya Satpute

Learn Ansible Automation: A Beginner to Expert Guide

Thank you so much for referring my notes! 🌟

I really appreciate your support! If you found them helpful, feel free to like, share, or leave a comment! Your feedback means a lot to me and helps me create even better content.

😊 Thanks again! 🙌

Divya Vasant Satpute