Ansible Cheat Sheet with Real-Time Use Cases - Part 1 🚀

Introduction

Ansible is a powerful open-source automation tool that simplifies the configuration management, application deployment, and task automation processes. This cheat sheet provides a quick reference to commonly used Ansible commands and concepts, accompanied by real-time use cases to illustrate their practical application.

1. Inventory Management

- **Command:** `ansible-inventory`
- **Use Case:** List all hosts in the inventory.

- `ansible-inventory --list`

2. Ad-hoc Commands

- **Command:** `ansible`
- **Use Case:** Run a command on all hosts in the inventory.

```
ansible all -m command -a "uptime"
```

3. Playbooks

- **Command:** `ansible-playbook`
- **Use Case:** Execute a playbook.

```
ansible-playbook deploy-app.yaml
```

4. Variables

- **Command:** `{{ variable_name }}`
- **Use Case:** Use a variable in a playbook.

```
tasks:
  - name: Ensure package is installed
    apt:
      name: "{{ package_name }}"
      state: present
```

5. Loops

- **Command:** `with_items`
- **Use Case:** Iterate over a list in a playbook.

```
tasks:
  - name: Create users
    user:
      name: "{{ item }}"
      state: present
    with_items:
      - user1
      - user2
```

6. Conditionals

- **Command:** `when`
- **Use Case:** Add a condition to a task.

```
tasks:
  - name: Restart Apache if config file changes
    service:
      name: apache2
      state: restarted
    when: "'apache.conf' in ansible.builtin.changed_files"
```

7. Roles

- **Command:** `ansible-galaxy`
- **Use Case:** Create a new role.

```
ansible-galaxy init my-role
```

8. Vault

- **Command:** `ansible-vault`
- **Use Case:** Encrypt sensitive data.

```
ansible-vault encrypt secrets.yaml
```

9. Dynamic Inventory

- **Command:** `ansible-inventory --graph`
- **Use Case:** Display the inventory structure in a graph.

```
ansible-inventory --graph
```

10. Handlers

- **Command:** `handlers`
- **Use Case:** Define a handler to restart a service only if a task triggers it.

```
handlers:
  - name: restart apache
    service:
      name: apache2
      state: restarted
```

11. Tags

- **Command:** `--tags` and `--skip-tags`
- **Use Case:** Run specific tasks or skip others based on tags.

```
ansible-playbook deploy-app.yaml --tags "install,config"
```

12. Ansible Vault Encryption and Decryption

- **Command:** `ansible-vault encrypt_string`
- **Use Case:** Encrypt sensitive strings for use in playbooks.

```
ansible-vault encrypt_string 'secret_password' --name 'db_password'
```

13. Custom Modules

- **Command:** `ansible-doc -l | grep your_module`
- **Use Case:** Develop and use custom Ansible modules.

```
ansible-doc -l | grep custom_module
```

14. Jinja2 Templating

- **Command:** `{{ variable | filter }}`
- **Use Case:** Use Jinja2 filters for dynamic content.

```
tasks:
  - name: Set dynamic variable
    set_fact:
      dynamic_value: "{{ base_value | upper }}"
```

15. Notify and Wait for Completion

- **Command:** `async` and `poll`
- **Use Case:** Execute tasks asynchronously and wait for their completion.

```
tasks:
  - name: Long-running task
    command: /path/to/long_running_script.sh
    async: 300
    poll: 0
```

```
    register: job_result

  - name: Wait for completion
    async_status:
      jid: "{{ job_result.ansible_job_id }}"
    until: job_result.finished
    retries: 30
```

Real-Time Use Cases

- **Deploying a Web Application:**
  - Use Ansible to deploy a web application on multiple servers, ensuring consistency and scalability.
- **Configuring Monitoring Agents:**
  - Automate the installation and configuration of monitoring agents on servers using Ansible playbooks.
- **Scaling Infrastructure:**
  - Dynamically scale your infrastructure by adding or removing servers based on demand using Ansible.
- **Continuous Integration/Continuous Deployment (CI/CD):**
  - Integrate Ansible into your CI/CD pipeline to automate deployment tasks and ensure efficient release management.
- **Database Configuration:**
  - Use Ansible to automate the setup and configuration of databases, ensuring a standardized environment.
- **Security Patching:**
  - Implement Ansible to automate the process of applying security patches across multiple servers simultaneously.
- **Automated Backup Strategy:**
  - Utilize Ansible to create an automated backup strategy, including regular backups and cleanup tasks.
- **Multi-Environment Deployment:**
  - Manage deployments across multiple environments (e.g., development, staging, production) using dynamic inventories and environment-specific playbooks.
- **Infrastructure Monitoring Setup:**
  - Automate the deployment of monitoring tools and configurations across servers to ensure real-time visibility into infrastructure health.
- **Custom Module for Application-specific Tasks:**
  - Develop a custom Ansible module to handle application-specific tasks, providing flexibility in automation.
- **Rolling Updates:**
  - Implement rolling updates for services, ensuring zero downtime during updates by using Ansible's serial and max_fail_percentage features.
- **Git Repository Management:**
  - Automate tasks related to Git repositories, such as cloning, pulling updates, and managing branches.

In the up-coming parts, we will discussion on more use cases & real time examples. So, stay tuned for the and follow @Prasad Suman Mohan for more such posts.

FOLLOW – Prasad Suman Mohan (for more updates)