Ansible Cheatsheet

A concise reference guide for Ansible, covering essential concepts, modules, and commands for effective infrastructure automation.

Core Concepts

Key Components

Control Node	The machine where Ansible is installed and from which plays are executed.
Managed Nodes	The servers or devices being managed by Ansible.
Inventory	A list of managed nodes, organized into groups. Can be a simple text file or a dynamic inventory script.
Modules	Reusable, standalone scripts that Ansible uses to perform tasks on managed nodes. Examples: `copy`, `file`, `apt`.
Tasks	A single unit of work defined in a playbook, calling a specific module with specific arguments.
Playbooks	YAML files that define a set of tasks to be executed on managed nodes. Orchestrates the configuration management process.

Idempotency

Ansible modules are designed to be idempotent, meaning they only make changes if necessary to bring the system to the desired state. This prevents unintended side effects from repeated playbook runs.

Ansible Configuration Files

`ansible.cfg`	Main configuration file for Ansible. Sets defaults for inventory location, module paths, etc. Can be located in `/etc/ansible/`, `~/.ansible.cfg`, or the current directory.

Playbooks & Syntax

Basic Playbook Structure

```yaml
---
- hosts: all
  become: true
  tasks:
    - name: Example Task
      module_name: module_options
```

- `hosts`: Specifies the target hosts or groups from the inventory.
- `become`: Escalates privileges (runs tasks as root).
- `tasks`: A list of tasks to be executed.

Common Modules

`apt`	Manages apt packages (install, remove, update).
`yum`	Manages yum packages (install, remove, update).
`copy`	Copies files to managed nodes.
`file`	Manages file attributes (permissions, ownership, symlinks).
`service`	Manages services (start, stop, restart, reload).
`user`	Manages user accounts.
`template`	Templates a file out to a remote system.

Variables

Variables can be defined in inventory files, playbook files, or as command-line arguments.

Example:

```yaml
vars:
  http_port: 8080

tasks:
  - name: Configure web server
    template:
      src: webserver.conf.j2
      dest: /etc/webserver.conf
      vars:
        port: "{{ http_port }}"
```

Commands & Usage

Common Commands

`ansible --version`	Displays the Ansible version.
`ansible-playbook playbook.yml`	Executes an Ansible playbook.
`ansible all -m ping`	Runs the `ping` module on all hosts in the inventory to check connectivity.
`ansible-galaxy install <role_name>`	Installs an Ansible role from Ansible Galaxy.
`ansible-vault encrypt <file>`	Encrypts a file using Ansible Vault.
`ansible-vault decrypt <file>`	Decrypts a file using Ansible Vault.

Inventory Management

Ansible uses an inventory file to define the managed nodes. The default location is `/etc/ansible/hosts`. You can specify a different inventory file using the `-i` option.

Example:

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

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

Ad-Hoc Commands

Ad-hoc commands are a quick way to execute single tasks on managed nodes without writing a full playbook.

Example:

```
ansible webservers -m shell -a 'uptime'
```

This command executes the `uptime` command on all hosts in the `webservers` group.

Advanced Features

Roles

Roles are a way to organize and reuse Ansible content. A role typically includes tasks, variables, handlers, and templates.

Directory Structure:

```
my_role/
├── tasks/
│   └── main.yml
├── handlers/
│   └── main.yml
├── vars/
│   └── main.yml
├── templates/
│   └── ...
└── meta/
    └── main.yml
```

Handlers

Handlers are tasks that are only executed when notified by another task. This is useful for restarting services after a configuration change.

Example:

```yaml
tasks:
  - name: Update webserver config
    template:
      src: webserver.conf.j2
      dest: /etc/webserver.conf
    notify: Restart webserver

handlers:
  - name: Restart webserver
    service:
      name: apache2
      state: restarted
```

Loops

Loops allow you to repeat a task multiple times with different values.

Example:

```yaml
tasks:
  - name: Create users
    user:
      name: "{{ item }}"
      state: present
    loop:
      - user1
      - user2
      - user3
```