Ansible Essentials Cheatsheet

A concise guide to Ansible, covering essential concepts, commands, and best practices for automating infrastructure and application deployment. Includes playbook structure, module usage, and task management.

Getting Started with Ansible

Installation & Setup

Install Ansible:

```
pip install ansible
```

Verify Installation:

```
ansible --version
```

Configure Hosts File (`/etc/ansible/hosts`):
Define your managed hosts or groups.

```
[webservers]
192.168.1.101
192.168.1.102

[dbservers]
192.168.1.103
```

SSH Key Configuration:
Ensure passwordless SSH access to managed hosts.

```
ssh-keygen -t rsa -N '' -f ~/.ssh/id_rsa
ssh-copy-id user@192.168.1.101
```

Basic Commands

`ansible -m ping all`	Verify connectivity to all managed hosts.
`ansible -m shell -a 'uptime' webservers`	Execute the `uptime` command on the `webservers` group.
`ansible-playbook playbook.yml`	Run an Ansible playbook.

Ansible Configuration File

The Ansible configuration file (`ansible.cfg`) allows you to customize Ansible's behavior. Common settings include:

- `inventory`: Path to the inventory file.
- `remote_user`: Default user for SSH connections.
- `private_key_file`: Path to the SSH private key.

```
[defaults]
inventory = /path/to/inventory
remote_user = ansible
private_key_file = ~/.ssh/id_rsa
host_key_checking = False
```

Playbook Essentials

Playbook Structure

A playbook is a YAML file containing one or more plays. Each play defines tasks to be executed on a set of hosts.

```
---  # Playbook Start
- hosts: webservers
  become: true  # Equivalent to sudo
  tasks:
    - name: Ensure Apache is installed
      apt:
        name: apache2
        state: present
    - name: Start Apache service
      service:
        name: apache2
        state: started
```

Tasks

`name`	A descriptive name for the task.
`module`	The Ansible module to be executed (e.g., `apt`, `service`, `copy`).
`args`	Parameters for the module.
`become`	Escalate privileges (sudo).
`become_user`	Specify the user for privilege escalation.
`register`	Store the task's output in a variable.

Handlers

Handlers are tasks that are only run when notified by another task. They are typically used for restarting services after configuration changes.

```
tasks:
  - name: Modify Apache config
    copy:
      src: httpd.conf
      dest: /etc/apache2/apache2.conf
    notify: Restart Apache

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

Variables & Templates

Defining Variables

Variables can be defined in several places:

- **Inventory file:** Host-specific or group-specific variables.
- **Playbook:** Using the `vars` section.
- **Included files:** Using `vars_files`.
- **Command line:** Using the `-e` option.

```
vars:
  http_port: 80
  max_clients: 200

tasks:
  - name: Set Apache port
    lineinfile:
      path: /etc/apache2/ports.conf
      regexp: '^Listen '
      line: 'Listen {{ http_port }}'
```

Variable Precedence

Ansible uses a specific order of precedence when resolving variables. From highest to lowest:

1. Command line (`-e`)
2. Role variables (`roles/role_name/vars/main.yml`)
3. Playbook `vars` section
4. Inventory group vars
5. Inventory host vars
6. `group_vars/all` and `host_vars/*` files

Templates

Templates allow you to dynamically generate configuration files using Jinja2 templating. They are useful for customizing configurations based on variables.

```
Listen {{ http_port }}
<VirtualHost *:{{ http_port }}>
  ServerName {{ ansible_hostname }}
  DocumentRoot /var/www/html
</VirtualHost>
```

```
tasks:
  - name: Deploy Apache virtual host
    template:
      src: vhost.conf.j2
      dest: /etc/apache2/sites-
available/000-default.conf
    notify: Restart Apache
```

Advanced Features

Roles

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

```
ansible-galaxy init my_role
```

Directory structure:

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

Includes

`include_tasks`	Include a list of tasks from another file.
`include_vars`	Include variables from another file.
`import_tasks`	Statically include a task list at playbook parse time.

Conditionals

Tasks can be conditionally executed using `when` clauses. This allows you to run tasks only when certain conditions are met.

```
tasks:
  - name: Install package on Debian
    apt:
      name: package_name
      state: present
    when: ansible_os_family == 'Debian'
```