Ansible Cheatsheet

Ansible Cheatsheet

Installation

```
# Install Ansible
pip install ansible-core  # Core only
pip install ansible        # Full with extras

# Verify installation
ansible --version
ansible-galaxy --version
```

Inventory Management

```
# hosts.ini
[webservers]
web01.example.com
web02.example.com
web03.example.com

[dbservers]
db01.example.com
db02.example.com

# \[all:vars\]
# ansible_user=ubuntu
# ansible_ssh_private_key_file=~/.ssh/id_rsa
# ansible_python_interpreter=/usr/bin/python3

# \[webservers:vars\]
# http_port=80
```

```
# inventory.yaml (YAML format)
webservers:
  hosts:
    web01.example.com:
      http_port: 80
    web02.example.com:
      http_port: 80
  vars:
```

```
    ansible_user: ubuntu

dbservers:
  hosts:
    db01.example.com:
    db02.example.com:

all:
  children:
    webservers:
    dbservers:
  vars:
    ansible_python_interpreter: /usr/bin/python3
```

Basic Commands

```
# Check connectivity
ansible all -i hosts.ini -m ping

# Run ad-hoc command
ansible webservers -i hosts.ini -m shell -a "uptime"

# Copy file
 ansible all -i hosts.ini -m copy -a "src=/tmp/file dest=/tmp/file"

# Install package
ansible webservers -i hosts.ini -m apt -a "name=nginx state=present"

# Gather facts
ansible all -i hosts.ini -m setup
```

Playbook Structure

```
# playbook.yaml
---
- name: Configure Web Server
  hosts: webservers
  become: yes
  vars:
    http_port: 80
    server_name: example.com

  tasks:
    - name: Update apt cache
      ansible.builtin.apt:
        update_cache: yes
        cache_valid_time: 3600

    - name: Install nginx
      ansible.builtin.apt:
        name: nginx
        state: present

    - name: Configure nginx
      ansible.builtin.template:
        src: nginx.conf.j2
```

```
        dest: /etc/nginx/nginx.conf
        owner: root
        group: root
        mode: '0644'
      notify: Restart nginx

  handlers:
    - name: Restart nginx
      ansible.builtin.service:
        name: nginx
        state: restarted
```

Common Modules

File Operations

```
- name: Create directory
  ansible.builtin.file:
    path: /var/www/html
    state: directory
    mode: '0755'
    owner: www-data

- name: Copy file
  ansible.builtin.copy:
    src: /local/file
    dest: /remote/file
    mode: '0644'

- name: Template file (Jinja2)
  ansible.builtin.template:
    src: config.j2
    dest: /etc/app/config.conf
    mode: '0644'

- name: Download file
  ansible.builtin.get_url:
    url: https://example.com/file.tar.gz
    dest: /tmp/file.tar.gz
    mode: '0644'
```

Package Management

```
# Debian/Ubuntu
- name: Install package
  ansible.builtin.apt:
    name: nginx
    state: present
    update_cache: yes

- name: Install multiple packages
  ansible.builtin.apt:
    name:
      - nginx
      - python3-pip
      - git
```

```
    state: present

- name: Remove package
  ansible.builtin.apt:
    name: nginx
    state: absent

# RHEL/CentOS
- name: Install package (yum)
  ansible.builtin.yum:
    name: nginx
    state: present

# Alpine
- name: Install package (apk)
  community.general.apk:
    name: nginx
    state: present
```

Service Management

```
- name: Start service
  ansible.builtin.service:
    name: nginx
    state: started
    enabled: yes

- name: Restart service
  ansible.builtin.service:
    name: nginx
    state: restarted

- name: Check service status
  ansible.builtin.service:
    name: nginx
    state: started
  check_mode: yes
```

System Administration

```
- name: Create user
  ansible.builtin.user:
    name: deploy
    shell: /bin/bash
    system: yes
    generate_ssh_key: yes

- name: Add SSH key
  ansible.posix.authorized_key:
    user: deploy
    key: "{{ lookup('file',('~/.ssh/id_rsa.pub')) }}"
    state: present

- name: Setup cron job
  ansible.builtin.cron:
    name: "Daily backup"
```

```
    minute: "0"
    hour: "2"
    job: "/usr/local/bin/backup.sh"
```

Variables

```
# Define in playbook
vars:
  app_name: myapp
  app_version: 1.0.0
  servers:
    - server1
    - server2

# Define in external file (group_vars/all.yml)
app_name: myapp
app_port: 8080

# Use with Jinja2
- name: Print variable
  ansible.builtin.debug:
    msg: "App: {{ app_name }} on {{ ansible_fqdn }}"

# Loop over list
- name: Create multiple users
  ansible.builtin.user:
    name: "{{ item }}"
    shell: /bin/bash
  loop:
    - user1
    - user2
    - user3

# Loop with dictionary
- name: Configure services
  ansible.builtin.service:
    name: "{{ item.key }}"
    state: "{{ item.value }}"
  loop:
    - { key: "nginx", value: "started" }
    - { key: "mysql", value: "started" }
```

Conditionals

```
- name: Install nginx on Debian/Ubuntu
  ansible.builtin.apt:
    name: nginx
    state: present
  when: ansible_os_family == "Debian"

- name: Install nginx on RHEL/CentOS
  ansible.builtin.yum:
    name: nginx
    state: present
  when: ansible_os_family == "RedHat"
```

```
# Multiple conditions
- name: Configure for production
  ansible.builtin.copy:
    src: prod.conf
    dest: /etc/app/config.conf
  when:
    - environment == "production"
    - ansible_fqdn is match("^prod.*")

# Check if file exists
- name: Check if config exists
  ansible.builtin.stat:
    path: /etc/app/config.conf
  register: config_file

- name: Create config if missing
  ansible.builtin.copy:
    src: default.conf
    dest: /etc/app/config.conf
  when: not config_file.stat.exists
```

Handlers

```
# Define handlers section
handlers:
  - name: Restart nginx
    ansible.builtin.service:
      name: nginx
      state: restarted

  - name: Reload systemd
    ansible.builtin.systemd:
      daemon_reload: yes

# Notify handler from task
- name: Update nginx config
  ansible.builtin.copy:
    src: nginx.conf
    dest: /etc/nginx/nginx.conf
  notify: Restart nginx

# Force handler to run
- name: Always notify
  ansible.builtin.debug:
    msg: "Triggering handler"
  notify: Restart nginx
  changed_when: true
```

Loops

```
# Simple loop
- name: Install packages
  ansible.builtin.apt:
    name: "{{ item }}"
    state: present
  loop:
```

```
    - nginx
    - python3-pip
    - git

# Loop with index
- name: Configure interfaces
  ansible.builtin.copy:
    content: "Interface {{ item.0 }}: {{ item.1 }}"
    dest: "/etc/interfaces/{{ item.0 }}"
  loop:
    - [0, "192.168.1.1"]
    - [1, "192.168.1.2"]

# Until loop
- name: Wait for service to be ready
  ansible.builtin.uri:
    url: "http://localhost:8080/health"
    status_code: 200
  register: result
  until: result.status == 200
  retries: 10
  delay: 5

# Dictionary loop
- name: Create users with specific properties
  ansible.builtin.user:
    name: "{{ item.key }}"
    shell: "{{ item.value.shell }}"
  loop: "{{ users | dict2items }}"
  vars:
    users:
      deploy:
        shell: /bin/bash
      backup:
        shell: /bin/bash
```

Error Handling

```
# Ignore errors
- name: Run command that may fail
  ansible.builtin.command: /tmp/dangerous-command
  ignore_errors: yes

# Rescue block (try/catch)
- name: Attempt deployment
  block:
    - name: Deploy application
      ansible.builtin.get_url:
        url: "{{ app_url }}"
        dest: /var/www/app.tar.gz
  rescue:
    - name: Deployment failed, cleanup
      ansible.builtin.file:
        path: /var/www/app.tar.gz
        state: absent
    - name: Notify failure
      ansible.builtin.debug:
        msg: "Deployment failed, performing rollback"
```

```
  always:
    - name: Always run cleanup
      ansible.builtin.debug:
        msg: "Cleanup completed"

# Failed_when condition
- name: Check service status
  ansible.builtin.command: systemctl status nginx
  register: service_status
  changed_when: false
  failed_when: "'active (running)' not in service_status.stdout"
```

Jinja2 Templates

```
# Variables in template
{{ app_name }}
{{ app_port }}
{{ ansible_default_ipv4.address }}

# Conditionals
{% if environment == "production" %}
server_name: {{ prod_domain }}
{% else %}
server_name: {{ dev_domain }}
{% endif %}

# Loops
{% for user in allowed_users %}
allow {{ user }};
{% endfor %}

# Filters
{{ nginx_config | to_nice_json }}
{{ list_of_items | join(', ') }}
{{ '  some text  ' | trim }}
{{ 'HELLO' | lower }}
{{ array | unique | list }}
```

Useful Filters

```
- name: Show memory usage
  ansible.builtin.debug:
    msg: "{{ ansible_memtotal_mb }} MB total, {{ ansible_memfree_mb }} MB free"

- name: Create list from string
  ansible.builtin.set_fact:
    packages_list: "{{ packages_string | split(',') }}"

- name: Get unique values
  ansible.builtin.set_fact:
    unique_users: "{{ users | unique | list }}"

- name: Check if item in list
  ansible.builtin.debug:
    msg: "Package installed"
  when: package_name in installed_packages
```

```
- name: Parse YAML string
  ansible.builtin.set_fact:
    config: "{{ config_string | from_yaml }}"
```

Best Practices

1. **Use YAML format** for inventory (easier to read and maintain)
2. **Become (sudo)** only when needed, with `become: yes`
3. **Idempotence** - tasks should be safe to run multiple times
4. **Separate sensitive data** using Ansible Vault
5. **Use handlers** for service restarts (only restart when changed)
6. **Validate** with `--check` and `--diff` before applying
7. **Structure** with roles for larger projects
8. **Document** playbooks with comments and descriptions
9. **Version control** all playbooks and configurations
10. **Test** in staging before production