Ansible Cheatsheet

Quick Reference

CLI, Playbooks, Module, Vault und Best Practices auf einen Blick.

Von **Admin-Cheatmeister**

CLI-Befehle

21

`ansible --version`	Zeigt installierte Ansible-Version und Config-Pfad.
`ansible all -m ping`	Ad-hoc-Befehl: Erreichbarkeit aller Hosts testen.
`ansible-playbook site.yml`	Führt ein Playbook aus.
`ansible-playbook site.yml --check`	Dry-Run, keine Änderungen anwenden.
`ansible-playbook site.yml --diff`	Zeigt Unterschiede bei Datei-/Config-Änderungen.
`ansible-playbook site.yml -i inventory.yml`	Explizites Inventory angeben.
`ansible-playbook site.yml --syntax-check`	Nur Syntax prüfen, nicht ausführen.
`ansible-playbook site.yml --list-hosts`	Zeigt betroffene Hosts, ohne auszuführen.
`ansible-playbook site.yml --list-tasks`	Listet alle Tasks des Playbooks.
`ansible-playbook site.yml --start-at-task="Setup"`	Ausführung ab bestimmtem Task starten.
`ansible-galaxy init my_role`	Erstellt neue Rollen-Verzeichnisstruktur.
`ansible-galaxy install -r requirements.yml`	Installiert Rollen aus Requirements-Datei.
`ansible-galaxy collection install <name>`	Installiert eine Collection.
`ansible-galaxy role search <name>`	Sucht Rollen in Ansible Galaxy.
`ansible-doc -l`	Listet alle verfügbaren Module.
`ansible-doc apt`	Zeigt Dokumentation/Parameter zu einem Modul.
`ansible-inventory --list`	Inventory als JSON ausgeben.
`ansible-inventory --graph`	Inventory als Baumstruktur anzeigen.
`ansible-config list`	Alle Konfigurationsoptionen anzeigen.
`ansible-config dump`	Aktive Konfiguration (inkl. Defaults) anzeigen.
`ansible-config view`	Zeigt Inhalt der aktiven ansible.cfg.

Inventory

10

`[webservers]`	Definiert eine Host-Gruppe im INI-Format.
`web1 ansible_host=192.168.1.10`	Host mit expliziter IP-Zuweisung.
`[webservers:vars]`	Gruppenweite Variablen im INI-Format.
`[webservers:children]`	Gruppe aus anderen Gruppen zusammensetzen.
`ansible_user / ansible_port / ansible_ssh_private_key_file`	Verbindungs-Variablen pro Host.
`ansible_connection=local`	Lokale Ausführung statt SSH.
`group_vars/all.yml`	Variablen für alle Hosts zentral definieren.
`host_vars/web1.yml`	Host-spezifische Variablen in eigener Datei.
`aws_ec2 / azure_rm (Inventory-Plugins)`	Dynamische Inventories holen Hosts zur Laufzeit von Cloud-APIs.

`ansible-inventory -i inventory.yml --list`	Dynamisches/statisches Inventory validieren.

Playbook-Grundstruktur 11

`hosts: webservers`	Zielgruppe des Plays.
`become: true`	Privilege Escalation (sudo) aktivieren.
`gather_facts: true`	Facts der Zielhosts sammeln (Standard: an).
`vars:`	Play-lokale Variablen definieren.
`tasks:`	Liste auszuführender Module/Aktionen.
`handlers:`	Tasks, die nur bei notify laufen.
`roles:`	Wiederverwendbare Rollen einbinden.
`pre_tasks: / post_tasks:`	Tasks vor bzw. nach Rollen ausführen.
`vars_files:`	Externe Variablendateien einbinden.
`- import_playbook: other.yml`	Weiteres Playbook statisch einbinden.
`- include_tasks: file.yml`	Tasks dynamisch zur Laufzeit einbinden.

Wichtige Module 20

`apt: name=nginx state=present`	Paket unter Debian/Ubuntu installieren.
`yum: name=httpd state=latest`	Paket unter RHEL/CentOS installieren.
`package: name=git state=present`	Distributionsunabhängige Paketverwaltung.
`copy: src=... dest=...`	Datei unverändert auf Zielhost kopieren.
`template: src=file.j2 dest=...`	Jinja2-Template rendern und ablegen.
`file: path=... state=directory`	Verzeichnisse/Symlinks/Berechtigungen verwalten.
`service: name=nginx state=started enabled=true`	Dienst starten/stoppen und Autostart setzen.
`user: name=deploy state=present groups=sudo`	Benutzerkonto anlegen/verwalten.
`command: /usr/bin/uptime`	Befehl ohne Shell-Interpretation ausführen.
`shell: echo $HOME`	Befehl mit Shell-Features (Pipes, Variablen) ausführen.
`git: repo=... dest=... version=main`	Git-Repository klonen/aktualisieren.
`cron: name="backup" minute="0" hour="2" job="/usr/bin/backup.sh"`	Cronjob verwalten.
`lineinfile: path=... line=... regexp=...`	Einzelne Zeile in Datei sicherstellen/ändern.
`blockinfile: path=... block=...`	Textblock in Datei einfügen/verwalten.
`synchronize:`	Effizientes rsync-basiertes Kopieren.
`unarchive: src=... dest=...`	Archiv entpacken (optional remote holen).
`debug: msg="{{ var }}"`	Werte/Variablen zu Debug-Zwecken ausgeben.

`assert:`	Bedingungen prüfen, Playbook bei Fehlschlag abbrechen.
`wait_for: port=22`	Warten, bis Port/Bedingung erfüllt ist.
`uri: / get_url:`	HTTP-Requests bzw. Dateien per URL herunterladen.

Variablen & Facts 8

`gather_facts: true`	Sammelt Systemfakten (OS, IP, Hardware) zu Playbook-Start.
`ansible_facts['os_family']`	Zugriff auf gesammelte Facts.
`register: result`	Task-Ausgabe in Variable speichern.
`set_fact: my_var="value"`	Variable zur Laufzeit dynamisch setzen.
`vars_prompt:`	Interaktive Eingabe von Variablen zur Laufzeit.
`--extra-vars "env=prod"`	Variablen per CLI übergeben (höchste Priorität).
`ansible.builtin.setup`	Facts explizit mit dem setup-Modul abfragen.
`Variablenpriorität`	Grob: Rollen-Defaults < Inventory-Vars < group_vars/host_vars < Facts < Play-Vars < Rollen-Vars < registrierte Vars < Extra Vars (-e, immer höchste Priorität).

Bedingungen & Schleifen 7

`when: ansible_os_family == "Debian"`	Task nur bei erfüllter Bedingung ausführen.
`loop: [a, b, c]`	Moderne Schleife über eine Liste (empfohlen).
`with_items: [a, b, c]`	Ältere, klassische Schleifensyntax (Legacy).
`loop_control: loop_var=...`	Steuerung von Schleifenvariablen bei verschachtelten Loops.
`until: / retries: / delay:`	Wiederholung bis Bedingung erfüllt ist (Polling).
`failed_when: / changed_when:`	Eigene Logik für Fail-/Changed-Status definieren.
`ignore_errors: true`	Playbook bei Fehler in diesem Task nicht abbrechen.

Handlers & Notifications 5

`notify: restart nginx`	Löst Handler nach einer Task-Änderung aus.
`handlers:`	Definition eines Handlers im Play (Modul wie ein normaler Task).
`Handler laufen nur bei changed`	Werden nur ausgeführt, wenn ein Task tatsächlich etwas geändert hat.
`Handler laufen am Play-Ende`	Standardmäßig gebündelt nach allen Tasks, nicht sofort.
`meta: flush_handlers`	Erzwingt sofortige Ausführung anstehender Handler.

Roles 9

`ansible-galaxy init rollenname`	Erstellt Standard-Verzeichnisstruktur einer Rolle.
`roles/rollenname/tasks/main.yml`	Haupt-Tasks der Rolle.
`roles/rollenname/handlers/main.yml`	Handler der Rolle.
`roles/rollenname/defaults/main.yml`	Standardwerte (niedrigste Variablenpriorität).
`roles/rollenname/vars/main.yml`	Feste Rollenvariablen (hohe Priorität).
`roles/rollenname/templates/`	Jinja2-Templates der Rolle.
`roles/rollenname/files/`	Statische Dateien der Rolle.
`roles/rollenname/meta/main.yml`	Metadaten und Rollen-Abhängigkeiten.
`roles: - role: rollenname vars: {...}`	Rolle im Playbook einbinden, mit Parametern.

Templates (Jinja2) 9

`{{ variable }}`	Variable im Template ausgeben.

`{% if condition %} ... {% endif %}`	Bedingte Logik im Template.
`{% for item in list %} ... {% endfor %}`	Schleife im Template.
`{{ var | default('x') }}`	Filter: Fallback-Wert setzen.
`{{ var | upper }} / {{ var | lower }}`	Filter: Groß-/Kleinschreibung.
`{{ list | join(',') }}`	Filter: Liste zu String verbinden.
`{{ var | to_json }} / {{ var | to_yaml }}`	Filter: Datenformat-Konvertierung.
`{{ var | bool }} / {{ var | int }}`	Filter: Typumwandlung.
`template: (Modul)`	Rendert eine .j2-Datei und legt sie auf dem Zielsystem ab.

Ansible Vault 10

`ansible-vault create vars/secrets.yml`	Neue verschlüsselte Datei anlegen.
`ansible-vault edit vars/secrets.yml`	Bestehende verschlüsselte Datei bearbeiten.
`ansible-vault view vars/secrets.yml`	Inhalt anzeigen, ohne dauerhaft zu entschlüsseln.
`ansible-vault encrypt file.yml`	Klartextdatei verschlüsseln.
`ansible-vault decrypt file.yml`	Datei dauerhaft entschlüsseln.
`ansible-vault rekey file.yml`	Vault-Passwort ändern.
`ansible-vault encrypt_string 'wert' --name 'var'`	Einzelwert verschlüsseln, inline nutzbar.
`--ask-vault-pass`	Passwort interaktiv abfragen.
`--vault-password-file ~/.vault_pass`	Passwort aus Datei/Skript lesen.
`--vault-id label@prompt`	Mehrere benannte Vault-Passwörter verwenden.

Tags 5

`tags: ["deploy"]`	Task oder Rolle mit Tag versehen.
`ansible-playbook site.yml --tags "deploy"`	Nur getaggte Tasks ausführen.
`ansible-playbook site.yml --skip-tags "debug"`	Bestimmte Tags überspringen.
`--tags always`	Spezial-Tag: läuft immer, unabhängig vom Filter.
`ansible-playbook site.yml --list-tags`	Alle verfügbaren Tags im Playbook anzeigen.

Ad-hoc-Befehle 7

`ansible all -m ping`	Erreichbarkeit aller Hosts testen.
`ansible webservers -m command -a "uptime"`	Befehl auf einer Gruppe ausführen.
`ansible all -m setup`	Facts aller Hosts abrufen.
`ansible all -a "df -h" --become`	Befehl mit Root-Rechten ausführen.
`ansible webservers -m apt -a "name=nginx state=present" --become`	Paketinstallation als Ad-hoc-Befehl.
`ansible all -m copy -a "src=./f dest=/tmp/f"`	Datei ad hoc kopieren.
`ansible all -m user -a "name=test state=present"`	Benutzer ad hoc anlegen.

Wichtige CLI-Flags 11

`-i inventory`	Inventory-Datei/-Quelle angeben.

`-l webservers`	Ausführung auf Host-Gruppe/-Subset begrenzen.
`--check`	Dry-Run-Modus, keine echten Änderungen.
`--diff`	Zeigt Datei-/Konfigurationsdifferenzen.
`-v, -vv, -vvv, -vvvv`	Zunehmende Ausführlichkeit der Ausgabe (Debugging).
`--become (-b)`	Privilege Escalation aktivieren.
`-K (--ask-become-pass)`	Sudo-Passwort interaktiv abfragen.
`-u username`	SSH-Benutzer festlegen.
`-e "key=value"`	Extra-Variablen übergeben (höchste Priorität).
`-f 10`	Anzahl paralleler Forks (Hosts gleichzeitig).
`--step`	Interaktive Bestätigung pro Task.

Best Practices

10

`Idempotenz sicherstellen`	Module bevorzugen (statt command/shell), damit Playbooks wiederholt ausführbar bleiben.
`command/shell nur wenn nötig`	Kein natives Modul vorhanden? Dann creates/removes für Idempotenz nutzen.
`Secrets immer per Vault`	Niemals Klartext-Passwörter in Playbooks oder Repo.
`Rollen statt monolithischer Playbooks`	Bessere Wiederverwendbarkeit und Testbarkeit.
`--check --diff vor Produktivlauf`	Änderungen vorab validieren.
`Namenskonvention für Tasks`	Jeder Task erhält aussagekräftigen name: für Lesbarkeit und Logs.
`Versionskontrolle (Git)`	Für Playbooks/Rollen: Nachvollziehbarkeit und Rollback.
`ansible-lint verwenden`	Style- und Best-Practice-Checks vor Produktivbetrieb.
`Handler statt direkter Restarts`	Vermeidet unnötige/mehrfache Service-Neustarts.
`Variablen in group_vars/host_vars`	Statt hartcodiert: zentrale, umgebungsspezifische Konfiguration.