Fichier 1 : `ansible_cheatsheet.md`

Ansible cheatSheet

Création de l'environnement

`ssh-keygen` : Générer une paire de clés authentification.

- `-f` : Spécifier le nom du fichier de clé.
- `-t [ecdsa|rsa|dsa]` : Spécifier le type de clé (ex: rsa).
- `-b` : Spécifier la taille de la clé en bits.

`ssh-copy-id -i [FILE]` : Copier la clé publique vers un serveur distant.

`~/.ssh/authorized_keys` (côté client): Fichier où la clé est stockée sur le serveur distant.

YAML

Syntaxe

- `---` : Début du document YAML.
- `...` : Fin du document.
- `#` : Commentaire.
- `true`, `True`, `TRUE`, `yes`, `Yes`, `YES`, `on`, `On`, `ON` : Valeurs booléennes pour Vrai.
- `false`, `False`, `FALSE`, `no`, `No`, `NO`, `off`, `Off`, `OFF` : Valeurs booléennes pour Faux.

```
courses:
- pommes
- bananes
- lait
- œufs
```

-> Liste (Séquence)

```
personne:
nom: Dupont
prenom: Jean
age: 30
adresse:
  rue: Rue de la Paix
  ville: Paris
  code_postal: 75002
```

-> Dictionnaire (Clé: Valeur)

Commande Ansible

`ansible [OPTIONS]` : Exécuter une commande ad-hoc.

- `-i` : Spécifier le fichier d'inventaire.
- `-K` : Demander le mot de passe d'élévation de privilèges (sudo).
- `--become-user` : L'utilisateur cible pour l'exécution (ex: root).
- `--become-password-file` : Fichier contenant le mot de passe sudo.
- `-b` : Activer l'élévation de privilèges (become).
- `--list-hosts` : Lister les hôtes ciblés sans exécuter la commande.
- `--private-key` : Utiliser une clé SSH spécifique.
- `-m` : Module à utiliser (ex: ping, shell).
- `-a` : Arguments à passer au module.
- `--vault-id` : Identifiant pour déchiffrer des variables chiffrées.

Playbook

`ansible-playbook [OPTIONS] [YAML_FILE]` : Lancer un playbook.

- `-i` : Chemin vers l'inventaire.
- `-e` : Définir des variables supplémentaires (extra vars).
- `--syntax-check` : Vérifier la syntaxe sans exécuter.
- `--start-at-task [TASK_NAME]` : Commencer à une tâche précise.

- `--tags=[TAG_NAME(S)]` : Exécuter uniquement les tâches avec ces tags.
- `--list-tags` : Lister tous les tags disponibles.
- `--skip-tags=[TAG_NAME(S)]` : Sauter les tâches ayant ces tags.
- `--ask-vault-pass` : Demander le mot de passe Vault.
- `--vault-password-file=[PATH_FILE]` : Fichier contenant le mot de passe Vault.
- `--force-handlers` : Exécuter les handlers même si une tâche échoue.
- `--list-hosts` : Lister les machines ciblées.
- `--list-tasks` : Lister les tâches sans les exécuter.
- `--check` : Simulation (Dry Run), ne fait pas de changements.
- `--diff` : Afficher les différences de fichiers (utile avec --check).
- `--step` : Confirmer chaque tâche manuellement.
- `--vault-id` : ID pour le déchiffrement Vault.

Syntaxe

- `- name:` : Nom du play ou de la tâche.
- `- hosts:` : Machines cibles.
- `- vars:` : Définition de variables.
- `- vars_file` : Fichier de variables externe.
- `- tasks` : Liste des tâches à exécuter.
- `- become` : Activer sudo/root.
- `- pre_tasks` : Tâches exécutées avant les rôles.
- `- post_tasks` : Tâches exécutées après les tâches principales.
- `no_log` : Ne pas afficher le résultat (pour secrets).
- `register` : Enregistrer la sortie dans une variable.
- `- handlers` : Tâches déclenchées par une notification.

Exemple de syntaxe d'un playbook (YAML)

```
---
- name: Update web servers
  hosts: webservers
  remote_user: root

  tasks:
  - name: Ensure apache is at the latest version
    ansible.builtin.yum:
      name: httpd
      state: latest
  - name: Write the apache config file
    ansible.builtin.template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- name: Update db servers
  hosts: databases
  remote_user: root

  tasks:
  - name: Ensure postgresql is at the latest version
    ansible.builtin.yum:
      name: postgresql
      state: latest
  - name: Ensure that postgresql is started
    ansible.builtin.service:
      name: postgresql
      state: started
...
```

Tags par défaut

- `tagged` : Tâches ayant au moins un tag.
- `untagged` : Tâches sans tag.
- `always` : S'exécute toujours (sauf skip explicite).
- `never` : Ne s'exécute jamais (sauf appel explicite).

Exemple d'utilisation de tags dans un playbook

(Structure montrant : tags samba, tags installation, tags configuration)

Utilisation de rôles dans un playbook

- `roles:` : Liste des rôles à appliquer.
- `import_role` : Importer un rôle statiquement.
- `include_role:` : Inclure un rôle dynamiquement.
- `dependencies` : Dépendances définies dans `meta/main.yml`.

Conditions dans un playbook

- `when:` : Appliquer une condition booléenne à une tâche.

Exemple:

```
tasks:
- name: Shut down CentOS 6 and Debian 7 systems
  ansible.builtin.command: /sbin/shutdown -t now
  when: (ansible_facts['distribution'] == "CentOS" and
        ansible_facts['distribution_major_version'] == "6") or
        (ansible_facts['distribution'] == "Debian" and
        ansible_facts['distribution_major_version'] == "7")
```

Boucles

- `loop:` : Boucle standard (recommandée).
- `with_<lookup>` : Ancienne syntaxe générique.
- `until` : Répéter jusqu'à ce qu'une condition soit vraie (avec retries/delay).
- `with_items` : Boucle sur une liste simple.
- `with_lines` : Boucle sur les lignes de sortie d'une commande.
- `with_dict` : Boucle sur un dictionnaire (clé/valeur).
- `with_inventory_hostnames` : Boucle sur des hôtes de l'inventaire.
- `with_fileglob` : Boucle sur des fichiers correspondant à un motif.

`loop_control:` : Contrôler le comportement de la boucle.

- `loop_var` : Changer le nom de la variable (défaut: `item`).
- `retries` : Nombre d'essais (avec until).
- `delay` : Délai entre essais.
- `index_var` : Variable stockant l'index de l'itération.
- `{{ item }}` : Variable courante dans la boucle.

fichier d'inventaire

`ansible-inventory` : Outil pour afficher/vérifier l'inventaire.

- `--graph` : Afficher l'inventaire sous forme d'arbre.
- `-i [YAML_FILE]` : Spécifier le fichier source.
- `--export` : Exporter l'inventaire.
- `--output=[FILE_PATH]` : Fichier de sortie.
- `--list` : Sortie JSON complète.
- `/etc/ansible/hosts` : Emplacement par défaut.

Syntaxe

- `server[1:4]` : Range (server1 à server4).
- `[bruxelles]` : Définition de groupe.
- `[webservers:children]` : Groupe de groupes.
- `[webservers:vars]` : Variables de groupe.
- `[group_name]` : En-tête de groupe.
- `all` (groupe par défaut): Tous les hôtes.
- `ungrouped` (groupe par défaut): Hôtes sans groupe spécifique.

Variable

- `ansible_host=[HOSTNAME|IP]` : Adresse réelle de connexion.
- `ansible_port=[NUMBER]` : Port SSH.
- `ansible_user=[USERNAME]` : Utilisateur de connexion.
- `ansible_password=[PASSWORD]` : Mot de passe SSH (déconseillé).
- `ansible_connection=[CONNECTION_TYPE]` : ssh, local, winrm, docker.

- `ansible_ssh_private_key_file=[FILE_PATH]` : Clé privée spécifique.
- `ansible_shell_type=[SHELL]` : Shell cible (sh, csh, fish...).
- `ansible_python_interpreter=[PATH]` : Chemin python cible.
- `ansible_*_interpreter=[PATH]` : Interpréteur pour autres langages (perl, ruby...).

Fichier de configuration

`ansible.cfg` : Fichier de conf principal (ini).

Syntaxe

- `inventory=[FILE_PATH]` : Inventaire par défaut.
- `remote_port=[NUMBER]` : Port SSH par défaut.
- `remote_user=[USERNAME]` : Utilisateur par défaut.
- `ssh_private_key_file=[FILE_PATH]` : Clé SSH par défaut.
- `executable=[SHELL]` : Shell par défaut (ex: /bin/bash).
- `role_path=[PATH]` : Chemins de recherche des rôles.
- `vault_identity_list=vaultID@[FILE_PATH]` : Liste des fichiers de mot de passe vault.
- `force_handlers=True` : Force l'exécution des handlers en cas d'erreur.

Variables

- `./host_vars/<host_name>.yml` : Variables spécifiques à un hôte.
- `./group_vars/<group_name>.yml` : Variables spécifiques à un groupe.

Syntaxe

- `{{ var }}` : Afficher une variable.
- `{{ var ~ var2 }}` : Concaténation (tilde).
- `{{ var.primary.host }}` OU `{{ var['primary']['host'] }}` : Accès attributs dictionnaire.
- `{{ ansible_facts }}` (dictionnaire de variable spéciales): Infos système récoltées.

Roles

`./roles/` : Dossier local des rôles. `/etc/ansible/roles/` : Dossier système des rôles.

Exemple de structure

(Arborescence standard: defaults, templates, files, handlers, meta, tasks, vars)

- `defaults` : Variables par défaut (priorité faible).
- `files` : Fichiers statiques à copier.
- `handlers` : Gestionnaires d'événements (services).
- `templates` : Fichiers dynamiques (Jinja2).
- `meta` : Métadonnées et dépendances.
- `tasks` : Liste principale des actions.
- `vars` : Variables du rôle (priorité forte).

Ansible Vault

`ansible-vault [ACTION] [OPTIONS]` : Outil de chiffrement.

Liste d'actions

- `create` : Créer un fichier chiffré.
  - `--encrypt-vault-id=[ID_NAME]` : Utiliser un ID spécifique.
  - `--ask-vault-pass` : Demander le mot de passe.
  - `--vault-password-file=[FILE_PATH]` : Lire le mot de passe fichier.
- `modify` : Modifier le mot de passe d'un fichier chiffré (non standard, souvent `rekey`).
- `view` : Voir le contenu sans éditer.
- `encrypt` : Chiffrer un fichier existant.
  - `--encrypt-vault-id=[ID_NAME]` : ID spécifique.
- `decrypt` : Déchiffrer un fichier.
  - `--ask-vault-pass` : Prompt mot de passe.
  - `--vault-password-file=[FILE_PATH]` : Fichier mot de passe.
- `rekey` : Changer le mot de passe de chiffrement.
  - `--new-vault-id=[NEW_VAULT_ID]` : Nouvel ID.
  - `--new-vault-password-file=[FILE_PATH]` : Nouveau fichier mot de passe.

Ansible Galaxy

`ansible-galaxy role [ACTION]` : Gestion des rôles communautaires.

- `install [OPTIONS] [ROLE_NAMESPACE]` : Installer un rôle.
  - `--role-path` OU `-p` : Chemin d'installation.
- `list` : Lister les rôles installés.
- `remove [OPTIONS] [ROLE_NAMESPACE]` : Supprimer un rôle.
- `init [OPTIONS] [ROLE_NAME]` : Créer la structure d'un rôle.
  - `--init-path` : Chemin de création.
  - `-s` : (server) API server src.
  - `--type [ROLE_TYPE]` : Type (ex: container, apb).
  - `-e` : (extra-vars).
  - `-f` : Force.

`ansible-galaxy collection` : Gestion des collections.

Handlers

- `- handlers:` : Section de définition.
- `notify:` : Appelle le handler (dans tasks).
- `listen:` : Alias pour regrouper des notifications.
- `meta` : Tâches spéciales.
  - `flush_handlers` : Forcer l'exécution immédiate des handlers en attente.
- `force_handlers:` (Dans ansible.cfg ou cli) Ignorer les erreurs pour les handlers.

Module

`ansible-doc [MODULE]` : Documentation des modules.

- `-t` : Type de plugin (module, lookup, etc.).
- `-l` : Lister les modules disponibles.

Liste de modules

- `command` : Exécuter une commande (sans shell).
- `apt` : Gestion paquets Debian/Ubuntu.
- `yum` : Gestion paquets RHEL/CentOS.
- `ping` : Vérifier la connexion python.
- `package` : Gestionnaire de paquets générique.
- `copy` : Copier fichier local -> distant.
- `file` : Gestion fichiers/dossiers/perms.
- `template` : Rendu de template Jinja2.
- `service` : Gestion services (start/stop).
- `fetch` : Copier fichier distant -> local.
- `unarchive` : Décompresser une archive.
- `community.general.archives` : Gestion d'archives (module collection).
- `replace` : Remplacer texte via Regex.
- `tempfile` : Créer fichier temporaire.
- `add_host` : Ajouter un hôte à l'inventaire en mémoire.
- `debug` : Afficher des messages/variables.
- `setup` : Récupérer les facts système.
- `meta` : Actions de contrôle du playbook.
  - `flush_handlers` : Exécuter handlers.
  - `clear_host_errors` : Réinitialiser état erreur.
- `authorized_key` : Gérer clés SSH publiques.

Troubleshooting

`yamllint` : Linter pour la syntaxe YAML.

`ansible-lint` : Linter pour les bonnes pratiques Ansible.

ET options de la commande `ansible-playbook` (voir plus haut : `-vvv` , `--check`).

Gestion des erreurs

- `ignore_errors: true` : Continuer même si la tâche échoue.
- `ignore_unreachable:` : Ignorer si l'hôte est injoignable.
- `meta` :
  - `clear_host_errors` : Réactiver un hôte en échec.
- `failed_when:` : Définir condition d'échec custom.
- `changed_when:` : Définir condition de changement custom.
- `<var_name>.rc == 0` : Vérifier code retour (return code).
- `debugger: always` : Lancer le debugger en cas d'erreur.

Gestion des failures

`any_errors_fatal` : Arrête tout le playbook si un hôte échoue.

- `block:` : Groupe de tâches (try).
- `rescue:` : Tâches en cas d'erreur du block (catch).
- `always:` : Tâches toujours exécutées (finally).

Template Jinja2

- `{{ MyVar }}` : Affiche la variable.
- `{# Comment #}` : Commentaire (non affiché).
- `{% Expression% }` : Structure de contrôle (if, for).

```
 {% for ghost in groups['apt'] %}
{{ loop.index }} - {{ ghost }}
{% endfor %}
->
{% if db_host == "db.henallux.be" %}
# Action to be done
{% endif %}
->
{{ groups['apt'] | join(',') }} :
{{ user_password | password_hash('sha512') }} :
{{ nginx_port | default(80) }} :
```

Fichier 2 : `vagrant_cheatsheet.md`

Vagrant

Configuration

`vagrant init [options] [name [url]]` : Initialiser un répertoire Vagrant.

- `--box-version VERSION` : Spécifier la version de la box.
- `--output FILE` : Nom du fichier de sortie (Vagrantfile).
- `-f` : Écraser le fichier existant.
- `-h` : Aide.
- `-m` : (Minimal) Vagrantfile sans commentaires.

`vagrant provision [vm-name]` : Exécuter les provisionneurs configurés.

- `-h` : Aide.

`vagrant validate [options]` : Vérifier la syntaxe du Vagrantfile.

- `-h` : Aide.

VM manipulation

`vagrant up [options] [name|id]` : Démarrer la machine virtuelle.

- `--[no-]provision` : Forcer (ou empêcher) le provisionnement au démarrage.
- `-h` : Aide.

`vagrant halt [options] [name|id]` : Arrêter la machine (shutdown).

- `-f` : Force l'arrêt (coupure courant).
- `-h` : Aide.

`vagrant suspend [options] [name|id]` : Mettre en pause (sauvegarde RAM).

- `-a` : (all) Suspendre toutes les VMs.
- `-h` : Aide.

`vagrant resume [vm-name]` : Reprendre une VM suspendue.

- `--[no-]provision` : Relancer provisionnement ou non.
- `-h` : Aide.

`vagrant destroy [options] [name|id]` : Supprimer la VM.

- `-f` : Force sans confirmation.
- `-g` : (graceful) Tente un arrêt propre avant.
- `-h` : Aide.

Get informations

`vagrant status [name|id]` : État de la VM.

- `-h` : Aide.

Connection

`vagrant ssh [options] [name|id] [-- extra ssh args]` : Connexion SSH.

- `-c` : Exécuter une commande.
- `-h` : Aide.
- `-p` : (plain) Pas d'authentification/clés gérées par Vagrant.

Boxes management

`vagrant box add [options] <name, url, or path>` : Ajouter une box.

- `--box-version VERSION` : Version spécifique.
- `-a` : (add) Automatique.
- `-c` : (checksum) Vérifier l'intégrité.
- `-f` : Force.
- `-h` : Aide.

`vagrant box list [options]` : Lister les boxes installées.

- `-h` : Aide.
- `-i` : (in-use) Montrer si utilisée.

`vagrant box remove <name>` : Supprimer une box.

- `--box-version VERSION` : Version à supprimer.
- `-a` : (all) Toutes les versions.
- `-f` : Force.
- `-h` : Aide.

VagrantFile

```
Vagrant.configure("2") do |config|        # => Configuration version 2
config.vm.provider "virtualbox" do |vb|   # => Provider VirtualBox
vb.cpu = <int>                            # => Nombre de CPU
vb.gui = <boolean>                        # => Mode graphique ou headless
vb.memory = <int>                         # => Mémoire RAM
vb.name = <string>                        # => Nom de la VM
end

config.vm.hostname = <val>                # => Nom d'hôte
config.vm.network "forwarded_port", guest:<int>, host:<int>
# └> Redirection de port
```

`config.vm.network "private_network"` : Réseau privé (Host-Only).

- `[, type: <val>]` : dhcp ou static.
- `[, name: <val>]` : Nom interface.
- `[, ip: <val>]` : IP statique.
- `[, netmask: <val>]` : Masque sous-réseau.
- `[, virtualbox__intnet: <val>]` : Réseau interne VB.
- `[, auto_config: <boolean>]` : Config auto de l'IP.

`config.vm.network "public_network"` : Réseau public (Bridged).

- `[, ip: <val>]` : IP sur le LAN.
- `[, bridge: <val>]` : Interface physique pontée.
- `[, auto_config: <boolean>]` : Config auto.

`config.vm.synced_folder <val>, <val>` : Dossier partagé (Hôte, Invité).

- `[, disabled: <boolean>]` : Désactiver le partage.

`config.vm.provision <name>,` : Provisionneur Shell.

- `type: "shell"`
- `[, inline: <val>]` : Script dans le Vagrantfile.
- `[, run: "(once|always|never)"]` : Fréquence d'exécution.
- `[, after: <val>]` : Ordre exécution.
- `[, before: <val>]` : Ordre exécution.

`config.vm.provision <name>,` : Provisionneur File.

- `type: "file"`
- `source: <path>` : Fichier hôte.
- `dest: <path>` : Destination invité.

`config.vm.provision "(docker|podman)" do |d|` : Provisionneur Docker.

- `d.build_image <path>` : Construire image.
- `d.pull_image <name>` : Télécharger image.
- `d.run <name>` : Lancer conteneur.
- `[, cmd: <val>]` : Commande.
- `[, args: <val>]` : Arguments.
- `[, daemonize: <val>]` : Arrière-plan.
- `[, restart: <val>]` : Politique restart. `end` `end`

Fichier 3 : `containers_cheatsheet.md`

Registries

`login [OPTIONS] [SERVER]` : Se connecter à un registre.

- `-u USER` : Nom d'utilisateur.
- `-p PWD` : Mot de passe.

`/etc/containers/registries.conf` : Configuration des registres (Podman).

`[registries.search] registries = ["registry.access.redhat.com", "quay.io"]`

`[registries.insecure] registries = ['localhost:5000']`

`search [OPTIONS] TERM` : Chercher une image.

- `--limit` : Limiter le nombre de résultats.
- `--filter` : Filtrer (ex: is-official).

Images

`pull [OPTIONS] NAME[:TAG|@DIGEST]` : Télécharger une image.

`images [OPTIONS] [REPOSITORY[:TAG]]` : Lister les images.

- `-a` : Afficher toutes les images (y compris intermédiaires).
- `--format` : Formater la sortie (Go template/JSON).

`save [OPTIONS] IMAGE [IMAGE...]` : Sauvegarder image dans une archive.

- `-o` : Fichier de sortie (tar).

`load [OPTIONS]` : Charger image depuis archive.

- `-i` : Fichier d'entrée.

`rmi [OPTIONS] IMAGE [IMAGE...]` (only with podman): Supprimer image.

- `-f` : Forcer la suppression.
- `-a` : (all) Toutes les images.

`commit [OPTIONS] CONTAINER [REPOSITORY[:PORT]/]IMAGE_NAME[:TAG]` : Créer image depuis conteneur.

- `-a "" or --author ""` : Auteur.
- `--message ""` : Message de commit.
- `--format` : Format (docker ou oci).

`diff CONTAINER` : Inspecter changements système de fichiers.

`push [OPTIONS] IMAGE [DESTINATION]` : Envoyer image vers registre.

`tag [OPTIONS] SOURCE_IMAGE[:TAG] [REGISTRYHOST/][USERNAME/]TARGET_IMAGE[:TAG]` : Créer un tag (alias).

Réseaux

`network [options]` : Gestion réseaux.

- `ls` : Lister les réseaux.
- `inspect NETWORK` : Détails du réseau.
  - `-f` : Format.
- `create [options] NETWORK` : Créer un réseau.
  - `-d , --driver [bridge | host | macvlan| ipvlan]` : Type de réseau.
  - `--subnet=SUBNET` : Sous-réseau CIDR.
  - `--gateway=GATEWAY` : Passerelle.
  - `--ip-range` : Plage IP allocation.
  - `-o` : Options driver.
    - `com.docker.network.bridge.name=NAME` : Nom interface bridge.
    - `parent=INTERFACE` : Interface parente (macvlan).
  - `--internal` : Pas d'accès externe.
- `connect NETWORK CONTAINER` : Connecter conteneur au réseau.
  - `--ip` : Fixer IP.
- `disconnect NETWORK CONTAINER` : Déconnecter.
- `rm` : Supprimer réseau.
- `prune` : Supprimer réseaux inutilisés.

`port [CONTAINER] [PRIVATE_PORT[/PROTO]]` : Afficher mappage de ports.

Containers

`run [options] IMAGE [COMMAND] [ARG...]` : Créer et démarrer un conteneur.

- `-d` : Détaché (background).
- `-p` : Publier ports (Host:Container).
- `-l` : Label.
- `-t` : TTY (pseudo-terminal).
- `-i` : Interactif (stdin ouvert).
- `-e` : Variables d'environnement.
- `--name` : Nom du conteneur.
- `--rm` : Supprimer après arrêt.
- `-v` : Volume (Host:Container).
- `--ip` : IP fixe.
- `--network NETWORK` : Réseau à attacher.

`ps` : Lister conteneurs.

- `-a` : Tous (y compris arrêtés).
- `--format="{{.ID}} {{.Names}} {{.Status}}"` : Format sortie.

`exec [OPTIONS] CONTAINER COMMAND [ARG...]` : Exécuter commande dans conteneur actif.

- `options +- identiques à run` (-it, etc.)
- `-l` (podman only) : Latest (dernier conteneur).

`stop [OPTIONS] CONTAINER [CONTAINER...]` : Arrêter conteneur.

- `-a` : Tous.

`kill [OPTIONS] CONTAINER [CONTAINER...]` : Tuer conteneur (SIGKILL).

- `-s` : Signal spécifique.

`restart [OPTIONS] CONTAINER [CONTAINER...]` : Redémarrer.

`rm [OPTIONS] CONTAINER [CONTAINER...]` : Supprimer conteneur.

- `-f` : Force (si running).
- `-a` : Tous.

`logs [OPTIONS] CONTAINER` : Voir les journaux.

`cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH | CONTAINER:SRC_PATH DEST_PATH` : Copier fichiers.

`inspect [OPTIONS] NAME|ID [NAME|ID...]` : Infos JSON conteneur.

- `-f` : Format.
- `-f "{{range .Mounts}}{{println .Destination}}{{end}}" CONTAINER_NAME/ID` : Exemple filtre mounts.

Volumes

`volume`

- `create` : Créer volume.
- `inspect` : Voir détails.
  - `-f` : Format.
- `ls` : Lister.
  - `--format` : Format.
  - `-f` : Filtre.
- `prune` : Nettoyer inutilisés.
- `rm` : Supprimer.
  - `-f` : Force.

Directory setup when Selinux used in conjunction with Podman rootless

`podman unshare chown` : Changer proprio dans namespace user.

- `-R` : Récursif.
- `-v` : Verbeux.

`sudo semanage fcontext` : Gérer contextes fichiers SELinux.

- `-a` : Ajouter.
- `-t` : Type de contexte.

`sudo restorecon` : Appliquer contextes SELinux.

- `-R` : Récursif.

Container (Docker) files

File syntax

- `#` : Commentaire.
- `FROM [--platform=<platform>] <image>[:<tag>] [AS <name>]` : Image de base.
- `LABEL <key>=<value> <key>=<value> <key>=<value> ...` : Métadonnées.
- `RUN` : Exécuter commande au build.
  - `<command>` : Shell form.
  - `["executable", "param1", "param2"]` : Exec form.
- `EXPOSE <port> [<port>/<protocol>...]` : Ports écoutés.
- `ENV <key>=<value> ...` : Variables d'environnement.
- `COPY [--chown=<user>:<group>] <src>... <dest>` : Copie locale -> image.
- `ADD [--chown=<user>:<group>] <src>... <dest>` : Copie avancée (URL/tar).
- `USER <UID>[:<GID>]` : Changer d'utilisateur.
- `WORKDIR /path/to/workdir` : Répertoire de travail.
- `ARG` : Variables de build.
- `VOLUME ["/monvolume"]` : Point de montage.
- `ENTRYPOINT ["executable", "param1", "param2"]` : Commande principale.
- `CMD` : Arguments par défaut ou commande.
  - `["executable","param1","param2"]` : Exec form, préféré.
  - `["param1","param2"]` : Paramètres par défaut pour ENTRYPOINT.
  - `command param1 param2` : Shell form.

Building

`build [OPTIONS] PATH | URL | -` : Construire une image.

- `-t` : Tag (Nom:Tag).
- `-f` : Fichier Dockerfile spécifique.
- `--layers=false` (Podman only) : Désactiver le cache des couches.
- `--build-arg` : Passer variable ARG.

Fichier 4 : `cheatSheet_Docker_Ansible.md`

Ansible Docker cheatsheet

`community.docker` : Collection Ansible pour Docker.

Docker images

`community.docker.docker_image` : Gestion des images Docker (pull, push, build, tag).

- `name:` : Nom de l'image (ex: ubuntu, redis).
- `tag:` : Tag de l'image (défaut: latest).
- `pull:` : (Obsolète, utiliser source) Télécharger l'image.
- `push:` : Envoyer l'image vers le registre.
- `source: [build|load|pull|local]` : Méthode d'obtention de l'image.
- `archive_path:` : Chemin vers le fichier .tar (si source=load).
- `build:` : Options de construction.
  - `path:` : Chemin vers le contexte de build (contenant le Dockerfile).
  - `args:` : Arguments de build (build-args).
- `state: [present|absent]` : État souhaité de l'image.

`community.docker.docker_image_build` : Construire une image Docker spécifiquement.

- `name:` : Nom et tag de l'image finale.
- `path:` : Répertoire contenant le Dockerfile.
- `dockerfile:` : Nom du Dockerfile (si différent du défaut).
- `args:` : Arguments de build.

`community.docker.docker_image_load` : Charger une image depuis une archive tar.

- `path:` : Chemin vers le fichier .tar.

Return Values

- `return.image_names` : Liste des noms d'images chargés.

`community.docker.docker_image_info` : Récupérer les informations d'une image.

- `name:` : Nom de l'image à inspecter.

Return Values

- `return.images` : Liste des images trouvées.
  - `return.images.Created` : Date de création.
  - `return.images.ExposedPorts` : Ports exposés.
  - `return.images.Volumes` : Volumes définis.
  - `return.images.ExposedPorts` : Ports exposés.
  - `return.images.Config.Entrypoint` : Point d'entrée de l'image.
  - `return.images.RepoTags` : Liste des tags.
  - `return.images.RepoDigests` : Hashs (digests) de l'image.

`community.docker.docker_image_pull` : Télécharger une image depuis un registre.

- `name:` : Nom de l'image à télécharger.

`community.docker.docker_image_tag` : Taguer une image existante.

- `name:` : Nom de l'image source.
- `repository:` : Nouveau nom/repository complet.
- `existing_images: [keep|overwrite]` : Comportement si le tag existe déjà.

Docker Networks

`community.docker.docker_network` : Gérer les réseaux Docker.

- `name:` : Nom du réseau.
- `ipam_config:` : Configuration IPAM (adresses IP).
  - `aux_addresses:` : Adresses auxiliaires.
  - `gateway:` : Passerelle du réseau.
  - `iprange:` : Plage d'allocation IP.
  - `subnet:` : Sous-réseau (CIDR).
- `state: [present|absent]` : Créer ou supprimer le réseau.
- `driver: [bridge|overlay]` : Pilote réseau à utiliser.

`community.docker.docker_network_info` : Informations sur les réseaux.

- `name:` : Nom du réseau.

Return values

- `return.exists` : Booléen si le réseau existe.
- `return.network` : Dictionnaire des infos.
  - `return.network.Containers` : Conteneurs connectés.
  - `return.network.Created` : Date de création.

Docker Volumes

`community.docker.docker_volume` : Gérer les volumes persistants.

- `volume_name:` : Nom du volume.
- `state: [present|absent]` : Créer ou supprimer le volume.
- `driver_options:` : Options spécifiques au driver de stockage.

`community.docker.docker_volume_info` : Informations sur les volumes.

- `name:` : Nom du volume.

Return values

- `return.exists` : Booléen si le volume existe.
- `return.volume` : Infos du volume.
  - `return.volume.Name` : Nom.
  - `return.volume.Mountpoint` : Chemin sur l'hôte.

Docker Lifecycle Management

`community.docker.docker_container` : Gérer le cycle de vie des conteneurs.

- `name:` : Nom du conteneur.
- `image:` : Image à utiliser.
- `volumes:` : Liste des montages de volumes.
- `ports:` : Mappage de ports (Host:Container).
- `exposed_ports:` : Ports à exposer (informatif ou pour liens).
- `recreate:` : Forcer la recréation du conteneur.
- `restart_policy: [no|on-failure|always|unless-stopped]` : Politique de redémarrage.
- `detach:` : Lancer en arrière-plan (défaut: true).
- `interactive:` : Garder stdin ouvert (mode interactif).
- `restart:` : Redémarrer le conteneur si déjà démarré.
- `links:` : Lier à d'autres conteneurs (obsolète, préférer networks).
- `command:` : Commande à exécuter au lancement.
- `state: [present|absent|started|healthy|stopped]` : État souhaité.
- `env:` : Dictionnaire de variables d'environnement.
- `env_files:` : Fichier(s) contenant les variables d'env.
- `networks:` : Réseaux auxquels connecter le conteneur.
  - `name:` : Nom du réseau.
  - `ipv4_address:` : IP statique souhaitée.
  - `aliases:` : Alias réseau du conteneur.
- `network_mode: [bridge|host|none|container:<name>|default]` : Mode réseau.
- `mounts:` : Montages avancés (syntaxe type Swarm/Compose).
  - `type: [bind|npipe|tmpfs|volume|cluster|image]` : Type de montage.
  - `source:` : Source sur l'hôte ou nom du volume.
  - `target:` : Chemin dans le conteneur.
- `healthcheck:` : Configuration de vérification de santé.
  - `test:` : Commande de test.
  - `interval:` : Intervalle entre les tests.
  - `retries:` : Nombre d'essais avant échec.
  - `timeout:` : Temps max pour un test.

Example:

```
 - name: Create a redis container
   community.docker.docker_container:
     name: myredis
     image: redis
     command: redis-server --appendonly yes
     state: present
     recreate: true
     exposed_ports:
       - 6379

 - name: Start a container
   community.docker.docker_container:
     name: myapplication
     image: someuser/appimage
     state: started
     restart: true
     links:
       - "myredis:aliasedredis"
     devices:
       - "/dev/sda:/dev/xvda:rwm"
     ports:
       - "8080:9000"
       - "127.0.0.1:8081:9001/udp"
       - "8000-8100:9003"
       - "7000-7010:9010-9020"
     env:
       SECRET_KEY: "ssssh"
       BOOLEAN_KEY: "yes"
```

`community.docker.docker_container_info` : Infos sur un conteneur.

- `name:` : Nom du conteneur.

Return values

- `return.exists` : Booléen si le conteneur existe.
- `return.container` : Détails du conteneur.
    - `return.container.Mounts` : Points de montage.
    - `return.container.ExposedPorts` : Ports exposés.
    - `return.container.NetworkSettings.Networks.<network_name.IPAddress` : Adresse IP.
    - `return.container.HostConfig.PortBindings` : Ports mappés sur l'hôte.
    - `return.container.State.StartedAt` : Date de démarrage.

`community.docker.docker_container_copy_into` : Copier des fichiers dans un conteneur.

- `container:` : Nom du conteneur cible.
- `path:` : Chemin source sur l'hôte.
- `container_path:` : Chemin destination dans le conteneur.

`community.docker.docker_container_exec:` : Exécuter une commande dans un conteneur actif.

- `container:` : Nom du conteneur.
- `command:` : Commande à exécuter.
- `user:` : Utilisateur exécutant la commande.
- `chdir:` : Répertoire de travail pour la commande.

Return values

- `return.stdout` : Sortie standard.
- `return.stderr` : Erreurs standard.
- `return.rc` : Code de retour (0 = succès).

`community.docker.docker_prune` : Nettoyer les ressources inutilisées.

- `containers:` : Pruner les conteneurs stoppés.
- `images:` : Pruner les images inutilisées.
- `networks:` : Pruner les réseaux inutilisés.
- `volumes:` : Pruner les volumes inutilisés.
- `containers_filters:` : Filtres pour la suppression.
    - `until:` : Avant une certaine date/heure.
    - `label:` : Selon les labels.

Container connexion settings

`ansible_connection = community.docker.docker` (inventory entry): Se connecter via l'API Docker (pas SSH).

Docker compose Ansible

`community.docker.docker_compose_v2:` : Gérer des projets Docker Compose (plugin V2).

- `assume_yes:` : Ne pas demander confirmation.
- `project_name:` : Nom du projet (prefix des conteneurs).
- `project_src:` : Chemin du dossier contenant docker-compose.yml.
- `state: [present|absent|stopped|restarted]` État du projet.
- `services:` : Liste des services spécifiques à cibler.
- `files:` : Liste des fichiers compose (ex: docker-compose.prod.yml).
- `env_files:` : Fichiers d'environnement (.env).

Return Values:

- `return.actions` : Actions effectuées.
  - `return.actions.what` : Type d'action.
  - `return.actions.status` : Résultat.
- `return.containers` : Infos sur les conteneurs gérés.
  - `return.containers.State` : État.
  - `return.containers.Name` : Nom.
  - `return.containers.Service` : Service associé.
  - `return.containers.Image` : Image utilisée.
- `return.images` : Images construites/utilisées.

Example:

```
 - name: Tear down existing services
   community.docker.docker_compose_v2:
     project_src: flask
     state: absent

 - name: Create and start services
   community.docker.docker_compose_v2:
     project_src: flask
   register: output
```