Formation Ansible avancé — Travaux pratiques

Sébastien Vincent

2025

0 Prérequis

Tâche : préparation de l'environnement
Matériel : poste GNU/Linux, macOS ou Windows

0.1 Machines virtuelles

Les TPs vont utiliser des VMs GNU/Linux AlmaLinux 9.

- VM AWX sous Debian 12 : lawx
- VM contrôleur sous AlmaLinux 9 : lcontroller
- VM *web* pour l'environnement PROD : lsxbweb01 et lsxbweb02
- VM *database* pour l'environnement PROD : lsxbdb01 et lsxbdb02
- VM *web* pour l'environnement TEST : ltestweb01
- VM *database* pour l'environnement TEST: ltestdb01

Si vos machines **ne sont PAS** déjà provisionnées, suivez la procédure suivante :

- Vérifier dans le BIOS que les extensions de virtualisation sont activés
- Installer les logiciels suivants :
  - VirtualBox (`https://www.virtualbox.org`)
  - Vagrant (`https://www.vagrantup.com`)
- Récupérer l'archive :
  `https://framagit.org/s-vincent/training-ansible-advanced/-/archive/master/training-ansible-advanced-master.zip`
- Décompresser l'archive et aller dans le répertoire training-ansible-advanced-master :
  cd vagrant/
  vagrant up
- Vérifier que les VMs sont démarrées :
  vagrant status

0.2 Remarque

L'intégralité des TPs sera réalisé sur le poste GNU/Linux graphique ou la VM **lcontroller**. Dans le cas de la VM, il faudra se connecter à cette machine au préalable. Depuis le répertoire training-ansible-advanced/vagrant :
vagrant ssh lcontroller

1 TP 1

Tâche : Rôle myapache / mymariadb
Matériel : poste GNU/Linux ou Windows

1.1 Inventaire

- Créer un inventaire pour l'environnement de PROD avec la configuration suivantes :
  - web : lsxbweb01, lsxbweb02
  - db : lsxbdb01, lsxbdb02
- Créer un inventaire pour l'environnement de TEST avec la configuration suivantes :
  - web : ltestweb01
  - db : ltestdb01
- Toutes les machines ont les comptes / mot de passe suivants :
  - vagrant / vagrant
- Ecrire un playbook keys.yml ou une commande ad-hoc qui va
  - ajouter la clé publique SSH (~/.ssh/id_rsa.pub) du controleur pour le compte **vagrant** des serveurs cibles (groupe *all*)
- Tips : n'oubliez pas les options -k et -K pour le premier appel !
- Tips 2 : module *ansible.posix.authorized_key*
- Tips 3 : pensez à préciser l'utilisateur SSH (option *-u vagrant* ou ajouter dans la section vars *ansible_user: vagrant*)

1.2 Création d'un rôle myname.myapache

- Créer un rôle nommé **myname**.myapache (libre à vous de choisir le chemin)
- Cahier des charges :
  - installer le paquet httpd
  - copier les configurations (voir plus bas) dans /etc/httpd/conf.d/
  - si une des configurations change, redémarrer le service
  - s'assurer que le service httpd est bien démarré et activé
- Chaque environnement (PROD et TEST) a une variable myapache_websites de type liste contenant les informations de chaque site web (fqdn et path)
- Chaque site web :
  - doit être configuré via un fichier de configuration **séparé** (voir listing).
  - le paramètre DocumentRoot pointe vers un répertoire dédié dans /var/www/html/ (libre à vous de choisir le nom du répertoire)
  - copier un fichier index.html dans ce répertoire dédié avec le contenu "Hello from <fqdn>" (remplacer <fqdn> par le nom de domaine du site web)
- Sites web pour **l'environnement PROD** :
  - www.mycompany.lan
  - ww2.mycompany.lan

  - w3.mycompany.lan

- Sites web pour **l'environnement TEST** :

  - www.my-company.local
  - ww2.my-company.local
  - w3.my-company.local

Listing 1: myvhost.conf

```
<VirtualHost *:80>
    ServerName www.example.com
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html
</VirtualHost>
```

1.3 Création d'un rôle myname.mymariadb

- Créer un rôle nommé **myname**.mymariadb (libre à vous de choisir le chemin)
- Cahier des charges :
  - installer le paquet mariadb-server et python3-PyMySQL (respecter la casse !)
  - modifier le fichier /etc/my.cnf.d/mariadb-server.**cnf** pour que la section [mysql**d**] contienne le paramètre *bind-address* avec la valeur 0.0.0.0
    - https://gist.github.com/fevangelou/fb72f36bbe333e059b66
  - si la configuration change, redémarrer le service
  - s'assurer que le service mariadb est bien démarré et activé
- Chaque environnement (PROD et TEST) a une variable mymariadb_databases de type liste contenant les informations de chaque bases (name)
  - Pour **l'environnement PROD** : nextcloudprod, wordpressprod
  - Pour **l'environnement TEST** : testnextdloud, testwordpress
- Tips : utiliser les modules de la collection community.mysql.

1.4 Déploiement

- Créer un playbook apache.yml pour déployer le rôle **myname**.myapache sur le groupe web de chaque environnement
- Vérifier que tout fonctionne :
  curl -H "Host: www.mycompany.lan" lsxbweb01
  curl -H "Host: w3.mycompany.lan" lsxbweb02
  curl -H "Host: www.my-company.local" ltestweb01
  curl -H "Host: ww2.my-company.local" ltestweb01
- Créer un playbook mariadb.yml pour déployer le rôle **myname**.mymariab sur le groupe db de chaque environnement
- Vérifier que les bases de données sont présentes :
  ansible -m mysql_query -a "query='show databases;' login_unix_socket=/var/lib/mysql/mysql.sock" \
  -i inventory/prod/hosts -u vagrant -b db

1.5 Bonus : mise en place d'un répertoire protégé par mot de passe

- Ajouter une tâche qui installe le paquet python3-passlib sur les machines du groupe web
- Modifier le fichier template de virtualhosts pour ajouter un répertoire protégé par mot de passe (voir l'exemple dans le listing)
- Créer le répertoire *secure* **DANS le répertoire du site web**
- Ajouter un fichier index.html (avec le contenu "Top Secret") dans ce répertoire
- Les couples login / mot de passe suivants sont à créer dans un fichier vault chiffré (vault.yml)
  - user / password
  - user2 / password2
- Créer le fichier /etc/httpd/htpasswd avec le module *community.general.htpasswd* à partir des couples login / mot de passe
- Lancer le *playbook*
- Valider le résultat avec Firefox ou curl :
  curl -u user:password http://www.company.lan/secure/
  curl -u user:password http://ww2.company.lan/secure/

Listing 2: access.conf

```
Alias /secure /var/www/secure
<Directory /var/www/secure>
  Options indexes
  AuthName "Please authenticate"
  AuthType Basic
  AuthUserFile /etc/httpd/htpasswd

  require valid-user
</Directory>
```

1.6 Bonus : ajout des utilisateurs de la base de données

- Dans le rôle **myname**.mymariadb, ajouter les utilisateurs ayant accès aux bases de données
- Les environnements (PROD et TEST) ont une liste contenant les utilisateurs et les bases auxquels ils ont accès
  - arnold, accès aux bases nextcloudprod et wordpressprod
  - sylvester, accès aux bases testnextcloud et testwordpress
  - nextclouduser, accès à la base nextcloud de PROD et TEST
  - wpuser, accès à la base wordpress de PROD et TEST
- Note : l'accès aux bases est autorisé depuis n'importe quel IP (%)
- Note 2 : il y a bien une liste **PAR environnement**
- Tips : boucles imbriquées *with_subelements*

2 TP 2

Tâche : filtres, facts, ancres
Matériel : poste GNU/Linux ou Windows

2.1 Sites web / bases de données *extra*

- Dans le rôle myname.myapache :
  - ajouter une tâche qui va ajouter une liste de sites web additionnels facultative via une variable myapache_extra_websites
- Dans le rôle myname.mymariadb :
  - ajouter une tâche qui va ajouter une liste de base de données facultative via une variable mymariadb_extra_databases
- Attention : le rôle doit fonctionner même si la variable *extra* n'est pas définie
- Attention (2) : ne pas stocker la variable dans le **répertoire default** du rôle
- Tips : utiliser la factorisation et les filtres

2.2 Custom facts

- Créer un playbook facts.yml, qui va déployer le fichier codesite.fact sur le groupe *all* (voir listing) avec les permission d'exécution dans le répertoire /etc/ansible/facts.d
- Attention : le répertoire doit exister sinon il faut le créer
- Vérifier avec une commande ad-hoc ou un playbook que la variable codesite est bien remonté

Listing 3: codesite.fact

```
#!/usr/bin/env python3
import os
import json

hostname = os.uname().nodename
codesite = "UNKNOWN"

ret = {}

if hostname.startswith("ltest"):
  codesite = "TEST"
elif hostname.startswith("lsxb"):
  codesite = "PROD_STRASBOURG"

ret['codesite'] = codesite

print(json.dumps(ret))
```

2.3 Ancres et module_defaults

- Dans le rôle myname.myapache
  - modifier le fichier tasks/main.yml pour utiliser les ancres afin d'éviter la répétition des arguments *mode, owner, group* des modules ansible.builtin.file/copy/template
  - attention : il faudra surcharger si la valeur n'est pas celle par défaut (*state / mode* d'un répertoire vs un fichier)

- Dans le rôle myname.mymariadb
  - modifier le fichier tasks/main.yml pour utiliser module_defaults afin d'éviter la répétition des arguments *login_unix_socket, login_user, state* pour le module community.mysql.mysql_db/mysql_u

2.4 Delegate_to

- Dans le playbook apache.yml, ajouter une tâche déléguer au contrôleur (localhost) qui va vérifier que le port 80 est ouvert sur chaque serveur
- Dans le playbook mariadb.yml, ajouter une tâche déléguer au contrôleur (localhost) qui va vérifier que le port 3306 est ouvert sur chaque serveur

3 TP 3

Tâche : Assert et tests unitaires
Matériel : poste GNU/Linux ou Windows

3.1 Assert

- Dans le rôle mymariadb
- Ajouter une variable par défaut nommée mymariadb_listen_ip avec la valeur 127.0.0.1
- Modifier la tâche de configuration de l'adresse IP d'écoute avec cette variable
- **Avant cette tâche**, vérifier avec le module *ansible.builtin.assert* que cette variable contient bien une adresse IP valide
- Tester avec le playbook de déploiement en surchargeant la variable
- Faire de même avec la variable mymariadb_listen_port pour configurer le port d'écoute (défaut 3306) et vérifier qu'il est compris entre 1 et 65535 inclut
- Ajouter dans la tâche de configuration de l'adresse IP le port (en-dessous du paramètre *bind-address*) :
  port = {{ mymariadb_listen_port }}

3.2 Molecule myapache

- Dans le rôle myname.myapache
- Editer le fichier meta/main.yml et modifier le paramètre author pour avoir *author: myname*
- Ajouter des tests unitaires avec le framework Molecule
  - utilisation du driver docker
  - utilisation d'un conteneur docker.io/almalinux:9 + les spécificités systemd (voir slides)
  - déploiement du rôle + variables (liste des sites, ...) dans converge.yml
  - vérification dans verify.yml
    - **httpd.service** est bien *running / enabled*
    - le port 80 est bien ouvert (pas de delegate_to)
    - le bon contenu est affiché (FQDN) pour l'accès aux sites de PROD
      Tips : utiliser le module *ansible.builtin.uri* avec les arguments *headers* / return_content
- Dans le répertoire du rôle, tester avec *molecule converge* puis *molecule verify* et enfin *molecule idempotence*
- Dans le répertoire du rôle, tester avec *molecule test* qui va lancer l'intégralité du scénario (create, converge, verify, ...)

3.3 Molecule mymariadb

- Dans le rôle myname.mymariadb
- A la racine, ajouter un fichier requirements.yml pour ajouter la dépendance de la collections *community.mysql*
- Editer le fichier meta/main.yml et modifier le paramètre author pour avoir *author: myname*
- Ajouter des tests unitaires avec le framework Molecule
  - utilisation du driver docker

  - utilisation d'un conteneur docker.io/almalinux:9 + les spécificités systemd (voir slides)
  - déploiement du rôle + variables (liste des bases, ...) dans converge.yml
  - vérification dans verify.yml
    * **mariadb.service** est bien *running* / *enabled*
    * le port du service 3306 **sur localhost** est bien ouvert (pas de delegate_to)
    * la base de données est bien présente avec une requête SQL :
      select schema_name from information_schema.schemata where schema_name = 'NameOfTheDatabase'

- Dans le répertoire du rôle, tester avec *molecule converge* puis *molecule verify* et enfin *molecule idempotence*
- Dans le répertoire du rôle, tester avec *molecule test* qui va lancer l'intégralité du scénario (create, converge, verify, ...)

3.4 ansible-lint

- Installer ansible-lint :
  yum install -y python3.11 python3.11-pip
  python3.11 -m pip install ansible-lint
  ansible-lint --version
- Ajouter les fichiers .yamllint et .ansible-lint dans le répertoire formation/ ainsi que dans les rôles myname.myapache et myname.mymariadb
- Vérifier les fichiers YAML des playbooks
- Corriger les erreurs / avertissements
  - Ignorer ceux du fichier meta/main.yml

Listing 4: .yamllint

```
---
extends: default

rules:
  line-length:
    max: 150

  empty-lines:
    max-end: 1

  braces:
    min-spaces-inside: 0
    max-spaces-inside: 1

  brackets:
    min-spaces-inside: 0
    max-spaces-inside: 1

  truthy:
    allowed-values:
      - 'True'
      - 'False'
      - 'true'
      - 'false'
      - 'yes'
      - 'no'
```

Listing 5: .ansible-lint

```
---
# .ansible-lint
skip_list:
  - yaml[line-length]
  - yaml[comments-indentation]

warn_list:
  - package-latest
```

3.5 Bonus : argument_specs.yml

- Dans le rôle myname.myapache
  - ajouter la validation des arguments myapache_websites et myapache_extra_websites avec le fichier meta/argument_specs.yml du rôle
- Dans le rôle myname.mymariadb
  - ajouter la validation des arguments mymariadb_listen_ip, mymariadb_listen_port, mymariadb_databases et mymariadb_extra_databases avec le fichier meta/argument_specs.yml du rôle

4 TP 4

Tâche : Orchestration
Matériel : poste GNU/Linux ou Windows

4.1 Rolling-update

- Créer un *playbook* update_pkg.yml qui met à jour les paquets bash, nano et vim sur **toutes** les machines d'un environnement
- En *pré* installation, ajouter une tâche avec le module *ansible.builtin.debug* avec un message "Begin install"
- En *post* installation, ajouter une tâche avec le module *ansible.builtin.debug* avec un message "End install"
- Configurer le *play* pour déployer **l'ensemble des tâches** par lot de deux machines à la fois
- Configurer la tâche de post-installation pour être effectuer uniquement par lot **d'une machine à la fois**

4.2 Tâches asynchrones

- Créer un *playbook* docker.yml à déployer sur le groupe web :
- installer le paquet python3-requests
- installer le paquet **docker-ce**
  - repo : https://download.docker.com/linux/centos/9/$basearch/stable
  - clé GPG : https://download.docker.com/linux/centos/gpg
- s'assurer que le service **docker** est démarré et activé
- à l'aide de la collection *community.docker*, récupérer (pull) les images docker.io/nextcloud et docker.io/postgres de **manière asynchrone**
- tâche avec le module ansible.builtin.pause pour bloquer pendant 5 secondes
- attente bloquante de la fin du téléchargement des images Docker (60 essais avec 5 secondes entre chaque)

4.3 Mode pull avec ansible-pull

- Créer un playbook pull.yml qui va installer le paquet ansible sur toutes les machines
- Déployer le playbook sur l'environnement de TEST et de PROD
- Cloner le dépôt Git **http://**10.67.0.1**:3000**/utilisateur/playbooks.git
- Copier le même playbook update_pkg.yml dans le dépôt Git (**ne pas** changer le champs hosts)
- Copier le même fichier dans le dépôt Git sous le nom update_pull.yml
- Modifier le champs hosts du fichier update_pull.yml et mettre *hosts: localhost* (*all* prend tout le monde sauf localhost)
- Ajouter ces deux fichiers, commiter et pousser les changements dans Git
  - login : utilisateur
  - mot de passe : utilisateur

- Sur un des serveurs web, lancer la commande **en root**:
  ansible-pull -U **http://**10.67.0.1**:3000**/utilisateur/playbooks.git -C main update_pull.yml > "/root/update-pull-$(date +'%Y%m%d-%H%M').log"
- Vérifier le bon fonctionnement

4.4 Tâche planifiée et ansible-pull

- Dans le playbook pull.yml, ajouter une entrée CRON
  - utilisateur : root
  - période : tous les soirs à 22:00
  - commande :
    ansible-pull -U **http://**10.67.0.1**:3000**/utilisateur/playbooks.git -C main update_pull.yml > "/root/update-pull-$(date +'\%Y\%m\%d-\%H\%M').log"
  - (le caractère pourcent (%) est échappé car cron considère que c'est un retour à la ligne sinon)
- Déployer le playbook pull.yml sur les machines du groupe web (avec ansible-playbook)
- Vérifier la présence de la tâche planifiée avec la commande *crontab -e* (en étant root)

5 TP 5

Tâche : AWX
Matériel : poste GNU/Linux ou Windows

5.1 Prise en main

- Aller sur https://awx.example.lan
  - Login : admin
  - Mot de passe : lastawxhero
- Aller dans le menu templates
- Lancer le template de démonstration

5.2 Configuration équipes / utilisateurs

- Créer les équipes suivantes
  - 3P40
  - 3T40
- Créer les utilisateurs suivants et les ajouter dans l'équipe correspondante
  - Arnold Blackedecker, équipe 3P40
  - Sylvester Standing, équipe 3P40
  - Chuck Nourrice, équipe 3T40
  - Jicé Vayday, équipe 3T40

5.3 Configuration des *credentials*

- Ajouter un *credential* "prod" de type machine pour le login *vagrant* avec la clé SSH utilisée depuis le TP 1
- Ajouter un *credential* "test" de type machine pour le login *vagrant* avec la clé SSH utilisée depuis le TP 1
- Donner les droits d'utilisation du *credential* "prod" à l'équipe 3P40
- Donner les droits d'utilisation du *credential* "test" à toutes les équipes

5.4 Configuration des inventaires

- Dans AWX, créer un inventaire pour l'environnement de PROD avec les deux groupes web et db
- Rajouter les variables du groupe web (group_vars)
- Rajouter les variables du groupe db (group_vars)
- Rajouter les hôtes **avec leurs adresses IP** à la place du nom DNS
  - le déploiement se fait à travers un conteneur qui n'a pas accès au fichier /etc/hosts du controlleur!
  - dans la vraie vie, bien sûr que vous avez un serveur DNS pour faire la résolution de nom !
- Faire la même chose pour l'inventaire de l'environnement de TEST
- Donner les droits d'utilisation / lecture de l'inventaire de PROD à l'équipe 3P40
- Donner les droits d'utilisation / lecture de l'inventaire de TEST à toutes les équipes

5.5 Configuration des projets Git

- Dans AWX, créer un projet Git
  - nom : myplaybooks
  - type de Contrôle de la source : Git
  - URL de Contrôle de la source : http://10.67.0.1:3000/utilisateur/playbooks.git
  - branche : *main*
- Sauvegarder
- Donner les droits d'utilisation / lecture au projet Git à toutes les équipes
- Vérifier que le projet a réussi à se synchroniser (voir dans l'onglet Détails, statut du dernier Job)

5.6 Déploiement du playbook personnalisé sur l'environnement de prod

- Se logger avec l'utilisateur Arnold Blackedecker
- Créer un nouveau *modèle de job*
  - nom : Job prod
  - inventaire : prod
  - projet : myplaybooks
  - playbook : update_pkg.yml
  - informations d'identification : prod
- Sauvegarder
- Lancer le déploiement et vérifier les logs de sortie
- Relancer le déploiement en testant les options (verbosity, ...)

5.7 Déploiement du playbook personnalisé sur l'environnement de TEST

- Aller sur https://awx.example.lan et se logger avec l'utilisateur Chuck Nourrice
- Créer un nouveau *modèle de job*
  - nom : Job test
  - inventaire : test
  - projet : myplaybooks
  - playbook : async.yml
  - informations d'identification : test
- Sauvegarder
- Lancer le déploiement et vérifier les logs de sortie
- Relancer le déploiement en testant les options (verbosity, ...)