Projet Final : Déploiement d'une Application Web 3-Tiers avec Ansible

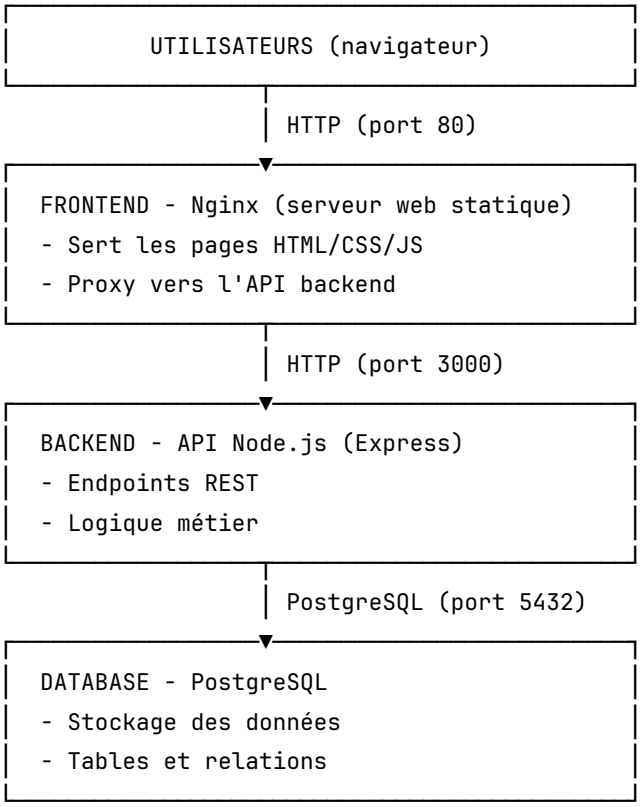
Objectifs Pédagogiques

Ce projet final vous permet de valider **toutes les compétences** acquises durant le workshop :

Créer et organiser des **playbooks** Ansible
Utiliser des **rôles** pour structurer le code
Gérer des **secrets** avec Ansible Vault
Déployer une **application multi-tiers** (frontend + backend + database)
Utiliser des **templates** Jinja2
Implémenter des **handlers** pour la gestion des services
Tester et valider un déploiement complet

Description du Projet

Vous allez déployer une **application web complète** composée de 3 couches :



Architecture Technique

- **Container `web1`** (SSH sur port local **2221**) : Frontend Nginx
- **Container `web2`** (SSH sur port local **2222**) : Frontend Nginx (haute disponibilité)
- **Container `db1`** (SSH sur port local **2223**) : Backend API + PostgreSQL

Note : On utilise 2 serveurs web pour simuler de la **haute disponibilité**.

Accès SSH aux conteneurs (OBLIGATOIRE pour la correction)

Pour que le projet soit corrigible, l'accès SSH doit être **standardisé**.

- **Utilisateur SSH imposé :** `root`
- **Mot de passe SSH imposé :** `ansible`
- **Hôte :** `127.0.0.1`
- **Ports :** `2221`, `2222`, `2223` (cf. docker-compose)

En authentification SSH par mot de passe, Ansible nécessite généralement `sshpass` côté machine de contrôle.

- macOS (Homebrew) : `brew install sshpass`
- Debian/Ubuntu : `sudo apt-get install -y sshpass`

Livrables Attendus

Vous devez rendre un projet avec cette structure **EXACTE** :

```
projet-ansible-final/
├── README.md
├── ansible.cfg
├── inventory/
│   └── hosts
├── group_vars/
│   └── all/
│       ├── vars.yml
│       └── vault.yml
├── playbooks/
├── roles/
│   ├── common/
│   │   └── tasks/
│   │       └── main.yml
│   ├── nginx/
│   │   ├── tasks/
│   │   │   └── main.yml
│   │   ├── templates/
│   │   │   └── nginx.conf.j2
│   │   └── handlers/
│   │       └── main.yml
│   ├── nodejs/
│   │   ├── tasks/
│   │   │   └── main.yml
│   │   ├── files/
│   │   │   ├── app.js
│   │   │   └── package.json
│   │   └── handlers/
│   │       └── main.yml
│   └── postgresql/
│       ├── tasks/
│       │   └── main.yml
│       ├── templates/
│       │   └── init.sql.j2
│       └── handlers/
│           └── main.yml
│
```

```
└── .vault_pass                # ⚠ NE PAS COMMITTER (pour tests locaux)
```

Exigences Techniques

Gestion du Vault Password

Utilisez le password prédéfini pour la correction : `ansible123`

Configuration requise

Inventaire (inventory/hosts) :

- Groupe `[webservers]` : web1 et web2 (ports SSH 2221 et 2222)
- Groupe `[appservers]` : db1 (port SSH 2223)
- Variables globales : `ansible_python_interpreter=/usr/bin/python3`

Variables (group_vars/all/vars.yml) :

- Variables **non sensibles** : configuration Nginx, Node.js, PostgreSQL
- Organisez vos variables de manière logique

Secrets (group_vars/all/vault.yml) :

- Variables **sensibles** à chiffrer : mots de passe PostgreSQL, clés API
- Chiffré avec le password `ansible123` (voir section Gestion du Vault Password ci-dessus)

Playbook Principal (playbooks/site.yml)

Votre playbook doit orchestrer le déploiement complet :

- Appliquer le rôle `common` sur **tous** les serveurs
- Appliquer le rôle `nginx` sur le groupe **webservers**
- Appliquer les rôles `postgresql` et `nodejs` sur le groupe **appservers**

Rôles à implémenter

Rôle `common` : Configuration de base

- Mise à jour du cache apt
- Installation de packages essentiels (curl, vim, htop, git)
- Optionnel : configuration du timezone

Rôle `nginx` : Frontend web

- Installer Nginx
- Créer une page HTML d'accueil (contenu **libre**, CV/portfolio/vitrine)
- Configurer Nginx avec proxy vers l'API backend (location `/api/` → `http://db1:3000/`)
- Implémenter des handlers pour redémarrage/rechargement

Rôle `postgresql` : Base de données

- Installer PostgreSQL
- Créer la base de données et l'utilisateur (mot de passe depuis `vault.yml`)
- Initialiser avec une table `users` (structure **libre**)

Rôle `nodejs` : Backend API

- Installer Node.js et npm

- Créer une API Express avec endpoints `/` et `/users`
- Connexion à PostgreSQL (utiliser variables d'environnement)
- Gérer les dépendances npm (package.json)

Critères de Validation (Barème /20)

Votre projet sera évalué selon les critères suivants :

1. Structure du Projet (/4)

- **[1 pt]** Structure de dossiers conforme (tous les dossiers/fichiers obligatoires présents)
- **[1 pt]** Fichier README.md complet et clair
- **[1 pt]** ansible.cfg correctement configuré
- **[1 pt]** Inventaire bien structuré avec groupes appropriés

2. Playbooks et Rôles (/6)

- **[2 pts]** Playbook `site.yml` fonctionnel et bien organisé
- **[1 pt]** Rôle `common` implémenté avec tâches de base
- **[1 pt]** Rôle `nginx` complet avec templates et handlers
- **[1 pt]** Rôle `postgresql` complet avec initialisation DB
- **[1 pt]** Rôle `nodejs` complet avec déploiement API

3. Gestion des Secrets (/3)

- **[1 pt]** Fichier `vault.yml` existant et chiffré
- **[1 pt]** Secrets utilisés correctement dans les rôles
- **[1 pt]** Mot de passe PostgreSQL non présent en clair dans le code

4. Tests Fonctionnels (/4)

- **[1 pt]** Syntaxe Ansible valide (`ansible-playbook playbooks/site.yml --syntax-check`)
- **[1 pt]** Déploiement fonctionnel (services démarrés, page web accessible, API accessible)
- **[2 pts]** Tests automatisés (playbook `playbooks/tests.yml`)
  - **[1 pt]** Playbook de tests présent et syntaxe valide (`ansible-playbook playbooks/tests.yml --syntax-check`)
  - **[1 pt]** Exécution des tests OK (`ansible-playbook playbooks/tests.yml`)

5. Qualité et Bonnes Pratiques (/3)

- **[2 pts]** `ansible-lint` passe sur les playbooks et `roles/`
- **[1 pt]** Utilisation de handlers pour les redémarrages de services

Exécution et Tests

Déploiement Complet

```
# 1. Vérifier la syntaxe
ansible-playbook playbooks/site.yml --syntax-check

# 2. Mode simulation (dry-run)
ansible-playbook playbooks/site.yml --check

# 3. Déploiement réel
ansible-playbook playbooks/site.yml

# 4. Rejouer uniquement si nécessaire (idempotence)
ansible-playbook playbooks/site.yml
```

Tests de Validation

```
# Tests automatisés (recommandé)
ansible-playbook playbooks/tests.yml

# Test 1 : Page web accessible (depuis les conteneurs)
ansible webservers -m uri -a "url=http://localhost status_code=200"

# Test 2 : API accessible (depuis db1)
ansible appservers -m uri -a "url=http://localhost:3000/ status_code=200"
ansible appservers -m uri -a "url=http://localhost:3000/users status_code=200"

# Test 3 : Process présents (containers sans systemd)
ansible webservers -m shell -a "pgrep -x nginx >/dev/null"
ansible appservers -m shell -a "pgrep -fa postgres >/dev/null"
ansible appservers -m shell -a "pgrep -f 'node.*app.js'"
```

README.md Obligatoire

Votre README doit contenir **au minimum** ces sections :

- **Description** : Brève présentation du projet
- **Architecture** : Schéma ou description de l'architecture 3-tiers
- **Prérequis** : Ansible, Docker, containers
- **Installation** : Étapes pour déployer le projet
- **Utilisation** : Comment accéder au frontend et à l'API
- **Tests** : Commandes pour valider le déploiement
- **Auteur** : Votre nom

Conseils et Astuces

Idempotence

Votre playbook doit être **idempotent** : - 1ère exécution : installe et configure tout - 2ème exécution : détecte que tout est OK, ne change rien

Points d'Attention

Ce qui est INTERDIT

- Mots de passe en clair dans le code
- Services non sécurisés (ports ouverts inutilement)
- Code dupliqué entre les rôles

Ce qui est ATTENDU

- Structure de dossiers respectée
- Variables bien organisées (vars.yml vs vault.yml)
- Handlers utilisés pour les redémarrages
- Templates Jinja2 pour les fichiers de configuration
- Documentation claire dans le README

Liberté Créative

Vous avez **carte blanche** sur :

- **Design de la page HTML** : CSS, animations, contenu
- **Fonctionnalités de l'API** : endpoints supplémentaires
- **Structure de la base de données** : tables additionnelles
- **Noms de variables** : tant que cohérents
- **Contenu du README** : format, style

Ce qui est évalué : La technique Ansible, pas le design graphique !

Rendu

Format : Archive `.tar.gz` ou lien Git (repository public)
Nom du fichier : `ansible-projet-[votre-nom].tar.gz` **Date limite :** 15 février 2026 23h59

Ressources

- Documentation Ansible
- Ansible Galaxy - Exemples de rôles
- Documentation PostgreSQL
- Documentation Express.js
- Documentation Nginx

Bon courage !