Projet Final : Industrialisation & Automatisation "Zero Touch"

Niveau : Master 2 Expert

Mode : Binôme

Durée : 4h

Livrable : Dépôt GitHub Public

1. Mise en situation

Vous disposez désormais d'une architecture microservices complète. Vous avez validé la sécurité de vos images (scan de vulnérabilités) et automatisé leur publication sur le GitHub Container Registry (GHCR).

L'objectif est maintenant d'**automatiser la mise en production** de cette stack, en respectant une contrainte majeure : **"No SSH"**.

NB : Pour la réalisation de ce projet, vous pouvez choisir de repartir du code de l'étudiant A, de l'étudiant B, ou d'utiliser la correction fournie en ligne.

Aucun humain ne doit se connecter au serveur pour le configurer. Toute l'infrastructure et le déploiement applicatif doivent être pilotés automatiquement depuis une pipeline CI/CD.

Votre mission : Concevoir et implémenter une chaîne de déploiement continu qui provisionne une infrastructure éphémère sur AWS, la configure, et y déploie la dernière version de l'application, le tout déclenché par un simple clic.

2. Architecture Cible

Vous devez automatiser le déploiement de l'architecture suivante :

- **Cloud Provider :** AWS (Région eu-west-3).
- **Compute :** Instance unique (type adapté au Free Tier).
- **OS :** Dernière version LTS d'Ubuntu (récupération dynamique).
- **Network/Security :**
  - Exposition stricte des ports nécessaires à l'application et à l'administration.
  - Le Frontend et l'API doivent être accessibles publiquement.
  - SSH ouvert uniquement pour l'outil de configuration (Ansible).
- **Application :** Stack Docker Compose (images stockées sur un registre privé).

3. Cahier des Charges Technique

Phase 1 : Infrastructure as Code (Terraform)

Responsable : Membre A

Vous devez produire le code Terraform nécessaire pour instancier l'environnement.

- **Contrainte 1 :** L'AMI ne doit pas être hardcodée (recherche dynamique via data source).
- **Contrainte 2 :** La clé SSH d'administration doit être générée à la volée par Terraform (pas de clé statique stockée dans le repo).
- **Contrainte 3 :** Le script doit exposer en output les informations critiques nécessaires à l'étape suivante (IP publique, Clé privée).

Phase 2 : Configuration Management (Ansible)

Responsable : Membre B

Vous devez produire un ou plusieurs Playbooks Ansible pour transformer un serveur vierge en serveur de production.

- **Rôle :** Installation du runtime Docker et des dépendances.
- **Sécurité :** Authentification auprès du registre d'images privé (GHCR) via token.
- **Déploiement :** Transfert et démarrage de la stack applicative.
- **Contrainte :** L'inventaire doit être dynamique ou généré (pas d'IP en dur dans un fichier hosts).

Phase 3 : Pipeline d'Orchestration (GitHub Actions)

Responsable : Binôme

Vous devez créer un Workflow unique (deploy.yml) déclenchable manuellement (workflow_dispatch).

Le pipeline doit exécuter séquentiellement :

1. **Build & Publish :** Construction des images Docker et push vers le registre privé.
2. **Infrastructure Provisioning :** Application du code Terraform.
3. **Bridge :** Récupération sécurisée des secrets (Clé SSH, IP) générés par Terraform pour les transmettre à Ansible.
4. **Configuration & Deployment :** Exécution du playbook Ansible sur la nouvelle infrastructure.

4. Contraintes de Réalisation

1. **Secret Management :** Aucun secret (Credentials AWS, Token GitHub, Clé SSH) ne doit apparaître en clair dans le code ou les logs. Utilisez les mécanismes natifs de GitHub Secrets.

2. **Environnement Éphémère :** Pour cet exercice, nous acceptons que le fichier d'état Terraform (tfstate) soit local au runner (et donc perdu à la fin du job). Chaque déploiement créera une nouvelle infrastructure.
3. **Adaptation Applicative :** Vous devez adapter le docker-compose.yml de développement pour qu'il consomme les images du registre privé (GHCR) en production, au lieu de builder localement.

5. Indices & Coups de pouce (Pour ne pas rester bloqué)

Voici quelques pistes pour résoudre les défis techniques les plus courants :

- **Le Pont Terraform -> Ansible :**
  Comment passer l'IP créée par Terraform à Ansible ?
  - *Piste :* Utilisez terraform output -raw <nom_output> pour récupérer une valeur brute. Vous pouvez stocker cette valeur dans une variable d'environnement du runner ($GITHUB_ENV) ou générer un fichier d'inventaire inventory.ini à la volée avec une commande echo.
- **La Clé SSH "Volatile" :**
  Terraform va générer une clé privée. Pour qu'Ansible puisse l'utiliser, il faut l'écrire dans un fichier temporaire sur le runner.
  - *Attention :* Ansible refusera d'utiliser la clé si les permissions sont trop ouvertes. Pensez à faire un chmod 600 key.pem juste après sa création.
- **L'Inventaire Dynamique :**
  Ne cherchez pas à faire compliqué. Un simple fichier texte généré par le script CI suffit :
  echo "[web]" > inventory.ini
  echo "$SERVER_IP ansible_user=ubuntu ansible_ssh_private_key_file=key.pem" >> inventory.ini
- **Docker Login à distance :**
  Votre serveur AWS doit télécharger vos images privées. Il a besoin de vos identifiants GitHub.
  - *Piste :* Passez votre ${{ secrets.GITHUB_TOKEN }} et votre ${{ github.actor }} comme variables à Ansible via l'option --extra-vars, puis utilisez-les dans le module docker_login d'Ansible.
- **La vérification d'hôte SSH :**
  C'est la première fois que le runner se connecte à ce nouveau serveur AWS. SSH va demander confirmation ("Are you sure...?"). Cela bloque le script.
  - *Solution :* Ajoutez l'option -o StrictHostKeyChecking=no dans vos commandes SSH ou dans la variable ansible_ssh_common_args.

6. Critères d'Évaluation

Le projet sera validé si et seulement si :

1. **Pipeline Vert :** Le workflow s'exécute de bout en bout sans erreur.
2. **Application Accessible :** L'URL fournie en fin de déploiement permet d'accéder au Dashboard.
3. **Qualité du Code :** Découpage clair (infra/, ansible/, .github/).

7. Livrables

Un **Dépôt GitHub Public** contenant :

- Tout le code source (App + Infra + Config).
- Un README.md détaillé contenant :
  - L'architecture du projet.
  - Les prérequis pour lancer le pipeline (quels Secrets GitHub créer ?).