Mise en place d’une plateforme DevOps Conteneurisée

1. Objectif

Mettre en place une infrastructure DevOps complète, automatisée et sécurisée, permettant l’intégration, le déploiement, l’analyse de code, la supervision et la gestion des applications .NET/Angular, à l’aide de Vagrant, Docker et plusieurs outils open source.

2. Architecture technique

* **Virtualisation** : VM Ubuntu provisionnée via Vagrant et VirtualBox.
* **Conteneurisation** : Utilisation de Docker Compose pour orchestrer tous les services DevOps.
* **Sécurité** : Pare-feu UFW configuré, ports essentiels ouverts, sécurité Jenkins renforcée.

3. Outils et services déployés

| **Service** | **Rôle** | **Port** |
| --- | --- | --- |
| Jenkins | Intégration continue (CI/CD) | 8080 |
| SonarQube | Analyse de qualité du code | 9000 |
| Grafana | Dashboards et visualisation | 3000 |
| Prometheus | Supervision et collecte de métriques | 9090 |
| Portainer | Gestion graphique des conteneurs Docker | 9443 |
| PostgreSQL | Base de données pour SonarQube | 15432 |
| SQL Server | Base de données SQL pour applications .NET | 11433 |
| Redis | Cache | 6379 |
| Seq | Centralisation des logs pour .NET | 5341 |
| Nginx | Reverse proxy (optionnel) | 80 |

4. Étapes principales réalisées

4.1 Installation des prérequis

* Installation de Git, Node.js, .NET SDK, Docker, Docker Compose, etc.
* Vérification des versions et de la disponibilité des outils.

4.2 Provisionnement de la VM

* Création et configuration d’une VM Ubuntu via Vagrant et VirtualBox.
* Attribution des ressources (CPU, RAM, réseau privé).

4.3 Déploiement des services DevOps avec Docker Compose

* Rédaction d’un fichier devops-compose.yml orchestrant tous les conteneurs nécessaires.
* Création de Dockerfiles personnalisés pour Jenkins (avec JDK 17 et .NET SDK).
* Lancement et arrêt des services de façon centralisée ou individuelle.

4.4 Configuration des outils globaux Jenkins

* Configuration des outils dans Jenkins : JDK, .NET SDK, NodeJS, Git, SonarQube Scanner, Docker.
* Vérification des chemins d’accès et des versions dans le conteneur Jenkins.

4.5 Sécurisation de la plateforme

* Activation et configuration du pare-feu UFW sur la VM.
* Ouverture stricte des ports nécessaires pour chaque service.
* Sécurisation de Jenkins (CSRF, gestion des utilisateurs, sauvegardes, etc.).

5. Points de contrôle & bonnes pratiques

* **Tous les services sont isolés dans des conteneurs Docker pour faciliter la maintenance et la portabilité.**
* **La gestion centralisée des outils via Jenkins permet d’automatiser les builds, tests et déploiements.**
* **La sécurité réseau est assurée par UFW, limitant l’exposition des ports.**
* **La configuration est reproductible et documentée (scripts de provisionnement, Dockerfiles, compose files).**

6. Utilisation & gestion

* **Démarrage/arrêt global** :  
  docker compose -f /vagrant/docker/devops-compose.yml up -d  
  docker compose -f /vagrant/docker/devops-compose.yml down
* **Démarrage/arrêt individuel** :  
  docker start <nom\_conteneur> / docker stop <nom\_conteneur>
* **Gestion des outils Jenkins** :  
  Via l’interface Jenkins > Gérer Jenkins > Configuration des outils globaux.
* **Sécurité** :  
  UFW actif, accès SSH maintenu, seuls les ports nécessaires sont ouverts.

7. Livrables

* Scripts d’installation et de configuration (Vagrantfile, Dockerfiles, scripts shell).
* Fichier Docker Compose complet.
* Documentation d’utilisation et de gestion des services.
* Plateforme DevOps opérationnelle, prête pour l’intégration et le déploiement des projets .NET/Angular.

**Ce cahier de charge résume l’ensemble de la démarche, la logique de sécurisation, l’automatisation et la gestion centralisée des outils DevOps sur la plateforme.**