

Cahier des Charges Kactus- Déploiement de services avec Docker et Docker Compose

I. Introduction

1. Présentation de la société

La société **Kactus** est spécialisée dans l'accompagnement des professionnels de la beauté. Elle propose des services variés allant de la gestion des réseaux sociaux à la création de sites web et à la conception de supports marketing.

2. Contexte du projet

Actuellement, les services informatiques de Kactus sont hébergés sur des machines virtuelles, ce qui engendre des coûts de maintenance élevée et complexifie les mises à jour et les évolutions. Pour répondre à ces enjeux, la DSI de Kactus souhaite moderniser son infrastructure système en migrant ses services existants vers une solution conteneurisée basée sur **Docker** et **Docker Compose**.

3. Objectifs du projet

La migration vers Docker vise à :

- **Réduire les coûts** en remplaçant les machines virtuelles par une solution plus légère et modulaire.
- **Faciliter les mises à jour et les déploiements** avec Docker Compose et l'automatisation via CI/CD.
- **Améliorer la scalabilité** en permettant une gestion dynamique des ressources.
- **Renforcer la sécurité** avec des solutions avancées de surveillance et de gestion des vulnérabilités.

4. Cible du projet

Le projet s'adresse principalement :

- Aux équipes internes de Kactus (développeurs, administrateurs systèmes, DSI).
- Aux prestataires externes qui interviennent sur l'infrastructure technique.

II. Description fonctionnelle

1. Fonctionnalités générales

Le projet couvrira les services suivants :

- Le site web WordPress de l'entreprise.
- L'application de gestion d'adresses IP **phpIPAM**.
- Des applications internes développées par Kactus.

2. Fonctionnalités détaillées et priorisées

Les fonctionnalités essentielles sont :

- Mise en place d'une infrastructure Docker stable et sécurisée.
- Gestion centralisée des conteneurs via **Portainer**.
- Création et déploiement d'images Docker personnalisées pour les services internes.
- Automatisation des mises à jour et des déploiements via un pipeline CI/CD.
- Supervision et gestion proactive des incidents grâce à des outils de monitoring avancés.
- Mise en œuvre de tests de sécurité renforcés (notamment via **Trivy**).

III. Contraintes et exigences

1. Contraintes techniques

- Utilisation des dernières versions stables de Docker, Docker Compose et Portainer.
- Mise en place de volumes persistants pour la conservation des données critiques.
- Adoption de normes de sécurité strictes (permissions limitées, gestion des accès SSH).
- Intégration de tests de sécurité dans le pipeline CI/CD pour détecter les vulnérabilités critiques.

2. Exigences fonctionnelles

- Chaque service déployé doit être accessible via les ports adéquats (HTTP, HTTPS, SSH).
- Les conteneurs doivent redémarrer automatiquement en cas de panne.
- Les données doivent être préservées lors des redémarrages des conteneurs.

3. Exigences de sécurité

- Mise en place de scans réguliers avec **Trivy** pour identifier les vulnérabilités critiques.
- Mise en place de règles strictes d'accès réseau pour limiter les risques d'intrusion.
- Gestion avancée des logs via **Loki** et configuration d'alertes automatisées.

IV. Solution technique envisagée

1. Outils et technologies utilisés

- **Docker Engine** et **Docker Compose** pour le déploiement des services conteneurisés.
- **Portainer** pour la gestion centralisée des conteneurs via une interface graphique.
- **Trivy** pour la gestion proactive des vulnérabilités.
- **Prometheus** et **Grafana** pour la supervision et la visualisation des métriques en temps réel.
- **Loki** pour la gestion des logs et la génération d'alertes avancées.
- **Outils d'IA** intégrés à Grafana pour l'automatisation de certaines tâches de surveillance.

2. Justification des choix techniques

- **Docker Compose** est privilégié pour sa simplicité et sa capacité à gérer les dépendances entre services.
- **Portainer** permet une administration graphique adaptée à tous les niveaux de compétences.
- **Prometheus** et **Grafana** apportent une visibilité avancée sur l'état du système et assurent un suivi temps réel.
- **Loki** permet de centraliser efficacement les logs tout en réduisant l'encombrement des volumes de stockage.

V. Stratégie de test

1. Tests fonctionnels

- Vérification du bon fonctionnement de chaque conteneur via docker ps.
- Vérification des connexions HTTP et HTTPS sur les services déployés.
- Tests de persistance des données après redémarrage des conteneurs.
- Simulation de pannes pour valider les mécanismes de reprise automatique.

2. Tests de sécurité

- Scan complet des images Docker via **Trivy** avec rapport détaillé des vulnérabilités identifiées.
- Tests d'intrusion simulés pour évaluer la résistance aux attaques externes.
- Vérification des permissions et des ports ouverts.
- Analyse du contenu des logs à la recherche de comportements suspects (via **Loki**).

3. Tests de performance et de charge

- Surveillance des indicateurs de performance (CPU, RAM, latence) via **Prometheus**.
- Analyse prédictive des anomalies à l'aide des outils IA intégrés à **Grafana**.

Livrables des tests :

- Rapport détaillé comprenant les résultats de chaque phase de test.
- Liste des correctifs et améliorations mises en œuvre en cas de défaillance.

VI. Échéancier et gestion du projet

| Phase | Durée estimée |
|-----------------------------------|---------------|
| Installation et Configuration | 0,5 jour |
| Création et Déploiement | 0,75 jour |
| Déploiement avec Docker Compose | 0,75 jour |
| Tests fonctionnels et de sécurité | 0,5 jour |
| Mise en place du Monitoring | 0,5 jour |

VII. Livrables finaux

1. Livrables du projet

- Infrastructure Docker stable et fonctionnelle avec tous les services requis.
- Documentation technique complète sur les procédures, configurations et tests réalisés.
- Tableaux de bord de supervision (Prometheus/Grafana) avec alertes configurées et intégration IA.
- Rapport de test final documentant la validation du bon fonctionnement et les actions correctives apportées.
- Plan de continuité et de reprise d'activité.

VIII. Conclusion

Ce cahier des charges vise à garantir une migration réussie vers une architecture conteneurisée moderne et sécurisée, répondant pleinement aux besoins stratégiques de la société **Kactus**. Un suivi rigoureux du planning et des tests permettra d'assurer la réussite du projet.