Parcours Ingénieur Système orienté DevOps

Projet de validation

Le groupe vétérinaire **Pet Clinic** (**PC**) possède une application locale (on-premises) assurant la prise de rendez-vous en ligne. Cette application utilise un serveur Web installé sur 4 serveurs physiques exécutant Linux Ubuntu Server 18.04. Elle utilise un autre serveur physique exécutant une base de données MySQL pour le stockage de données. La base de données contient des données :

- Les vétérinaires et leurs spécialités
- Les propriétaires des animaux et leurs coordonnées
- Les animaux et leurs rendez-vous de consultation

L'équipe de développement travaille activement sur la mise à jour de l'application. Vous pouvez cloner le dépôt Git du projet depuis le lien suivant :

https://github.com/HassenJRIDI/PetClinicWarFile

Les vétérinaires accèdent à la base de données en utilisant une section du site Web consacrée aux praticiens.

Les clients utilisent une section distincte du site Web pour la prise de rendez-vous.

Des problèmes récurrents affectent la disponibilité de l'application de prise de rendez-vous en ligne. Le service informatique de **PC** est limité en termes de ressources et a du mal à s'assurer que l'application est toujours disponible. Aussi, les équipes ne disposent d'aucune donnée pour vérifier si l'application fonctionne correctement après son déploiement. Nous avons besoin de mettre en place des systèmes de monitoring et de gestion de logs.

Le problème est devenu critique et l'application a connu une utilisation beaucoup plus importante que la normale, et le système a été rapidement surchargé et ne répond plus. L'équipe informatique a déterminé que le problème venait du manque de ressources des serveurs, mais il lui faut 6 semaines pour commander un nouveau matériel et créer des serveurs supplémentaires pour gérer la charge.

On vous demande de proposer une solution qui pourra résoudre tous ces problèmes.

Quelle que soit la solution que vous proposez, vous devez tenir compte du fait que le client souhaite que le cycle de vie de l'application soit exécuté de façon automatisée.

Vous avez décidé que **PetClinic (PC)** devrait se doter d'une usine d'intégration continue des développements en local.

1) Pour vendre l'idée à votre client (PC), vous allez devoir préparer un dossier de conception qui permettra de documenter l'architecture de la solution proposée ainsi que les arguments qui permettront de la soutenir.

Vous devez apporter des réponses aux questions suivantes :

- Quels sont les composants de la plate-forme d'intégration continue proposée ?
- La plate-forme d'intégration continue proposée pourrait-elle être étendue pour assurer un déploiement continu sur le cloud ?
- Préconisez-vous de conteneuriser l'application ?
- Quel type de service cloud recommanderiez-vous ?
- Comment justifieriez-vous votre choix par rapport aux problèmes rencontrés par l'équipe informatique ?
- Comment justifieriez-vous votre choix par rapport aux problèmes rencontrés par l'équipe de développement ?
- Comment pouvez-vous répondre aux exigences liées à la sécurité des données ?
- Comment s'assurer que l'application déployée sur le cloud peut bénéficier d'une haute disponibilité et qu'elle soit protégée contre les catastrophes qui pourraient survenir dans un centre de données (Data Center) ?
- Quels sont les composants du système de monitoring et de gestion de logs.

2) Dans une deuxième étape, la mise en place d'une usine de développement qui se basera essentiellement sur l'intégration et le déploiement continu.

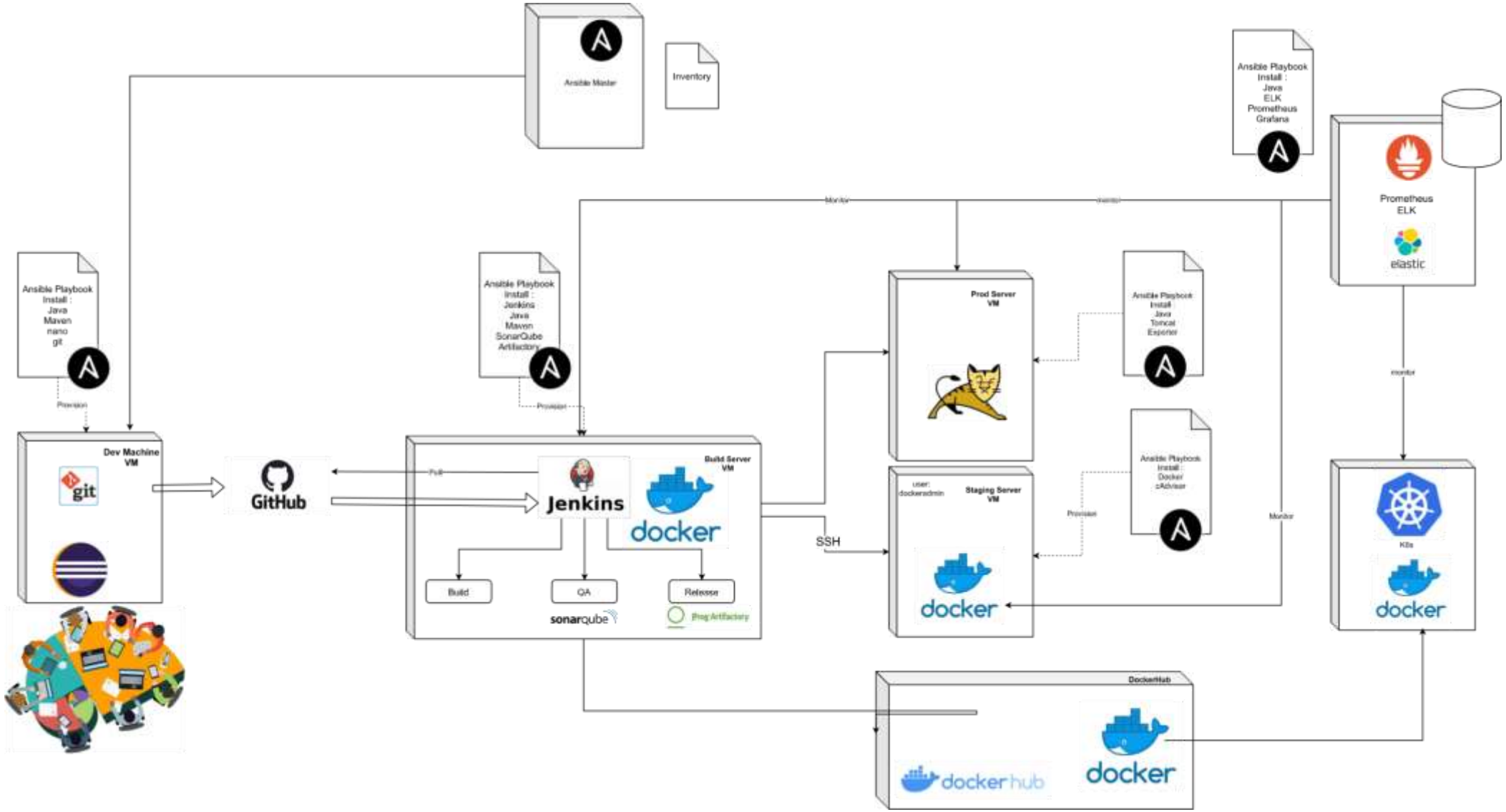
L'intégration continue sera la pièce maîtresse et le socle indispensable de ce projet. Que ce soit Jenkins ou Gitlab-CI, cela assurera la bonne compilation du code, le jeu des tests unitaires, le packaging, le déploiement et l'exécution des tests dans un environnement d'intégration.

L'intégration continue fiabilisera les activités de développement. Une fois en place, elle autorisera l'organisation du projet à pousser l'industrialisation au-delà des frontières de l'équipe de développement. Ainsi, le code packagé est ainsi archivé dans un référentiel central qui sera utilisée pour le déploiement sur l'ensemble des environnements, y compris la production.

Le déploiement automatisé, une nécessité, permettra d'automatiser progressivement les actions de déploiements auparavant réalisées manuellement par l'équipe d'exploitation (mise à jour de bases de données, déploiement dans un serveur d'applications, etc.).

Ces déploiements automatisés peuvent aller de scripts de déploiement dans les cas les plus simples jusqu'à la recréation complète de l'environnement cible (via des outils comme Docker et Kubernetes). Tout comme le code, ces scripts et paramétrages ont leur place dans un gestionnaire de sources.

Le schéma suivant illustre l'architecture à mettre en place.



Usine Logicielle DevOps