

ETIXI

CODE



Les Conteneurs: Une Révolution dans le Monde de l'Informatique

Plongeons dans l'univers passionnant des conteneurs, de Docker et des microservices - une transformation numérique qui bouleverse le secteur informatique.

 **by Etienne Koa**



Qu'est-ce qu'un Conteneur?

1

Unité d'Empaquetage

Les conteneurs permettent d'emballer une application avec toutes ses dépendances dans une unité standardisée et portable.

2

Environnement Cohérent

Chaque conteneur fournit un environnement d'exécution identique, assurant la reproductibilité et la fiabilité des applications.

3

Isolation et Sécurité

Les conteneurs offrent une isolation des ressources et une sécurité renforcée, limitant les risques de conflits et de failles.



Made with Gamma

Docker: Un Outil Révolutionnaire

Qu'est-ce que Docker?

Docker est un outil open-source qui permet de créer, déployer et exécuter des applications dans des conteneurs.

Avantages Clés

Portabilité, évolutivité, performance et simplicité d'utilisation font de Docker un outil incontournable pour les développeurs.

Écosystème Riche

Docker s'appuie sur une communauté active et un vaste écosystème d'outils et de services complémentaires.

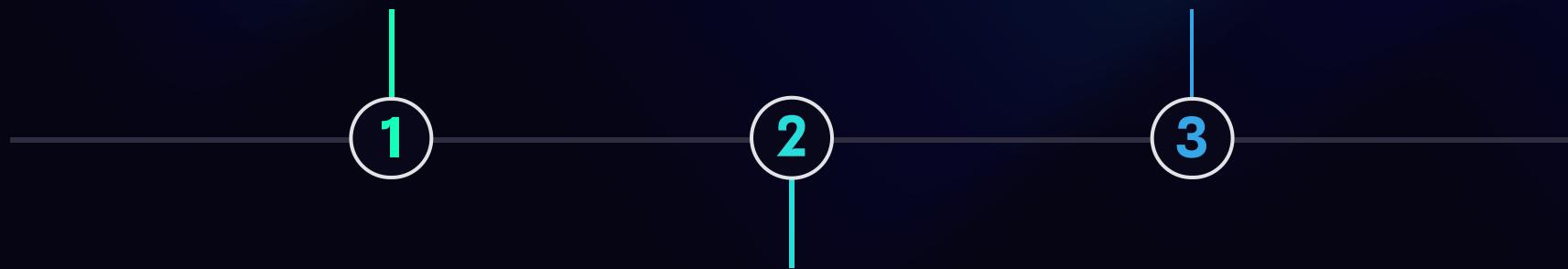
Manipuler les Conteneurs Docker

Créer

Construire une image Docker à partir d'un Dockerfile définissant l'environnement de l'application.

Gérer

Utiliser les commandes Docker pour interagir avec les conteneurs, les inspecter et les administrer.



Exécuter

Lancer un conteneur Docker à partir d'une image, en configurant les paramètres d'exécution.

Voici un résumé des points essentiels de la partie Docker Hands-on :

Introduction :

- **Objectif** : Cette partie est une introduction pratique à Docker.
- **Préparation** : Utilisation de Git Bash ou d'un terminal pour créer un répertoire et y placer un fichier Vagrant basé sur Ubuntu 20 pour installer Docker.

Étapes Clés :

1. Configuration de l'environnement :

- Utilisation de Vagrant pour configurer une machine virtuelle (VM) avec Docker.
- Commandes nécessaires disponibles dans la documentation Docker.

2. Installation de Docker :

- Lancer la VM avec `vagrant up`.
- Se connecter à la VM via SSH et vérifier l'état du service Docker (`systemctl status docker`).

3. Premier Test Docker :

- Exécution de la commande `docker run hello-world` pour tester Docker.
- Commandes utiles : `docker images` pour voir les images disponibles, `docker ps` pour voir les conteneurs en cours d'exécution, et `docker ps -a` pour tous les conteneurs.

4. Création et Gestion de Conteneurs :

- Exemple de création d'un conteneur Nginx avec des options de nom et de mappage de port (`docker run --name web01 -d -p 9080:80 nginx`).
- Accéder au conteneur via l'adresse IP de la VM et le port mappé.

5. Création d'Images Personnalisées :

- Création d'un Dockerfile pour définir une image basée sur Ubuntu.
- Construction de l'image avec `docker build -t <nom_image> ..`

6. Nettoyage :

- Commandes pour arrêter (`docker stop <nom_conteneur>`) et supprimer (`docker rm <nom_conteneur>, docker rmi <ID_image>`) les conteneurs et images pour préparer l'environnement pour la prochaine session.

Conclusion :

- **Documentation** : Toutes les commandes et le fichier Docker seront disponibles sur GitHub.
- **Préparation** : Nettoyez l'environnement avant la prochaine étape.

Le Projet Vprofile: Une Démonstration Pratique

Application Multicouche

Vprofile est une application web composée de microservices s'exécutant dans des conteneurs Docker.

Stockage de Données

Les données sont gérées par des bases de données Docker, garantissant la fiabilité et la scalabilité.

Orchestration avec Kubernetes

Kubernetes permet de déployer, gérer et mettre à l'échelle facilement les conteneurs Docker.

Déploiement Simplifié

Les conteneurs rendent le déploiement de l'application rapide, cohérent et indépendant de l'environnement.

Voici un résumé des points essentiels de la partie sur l'exécution de l'application Vprofile sur des conteneurs Docker :

Objectif et Contexte :

- **But** : Montrer comment exécuter l'application Vprofile sur des conteneurs Docker.
- **Aperçu** : Introduction pratique avant d'apprendre à tout configurer depuis zéro dans la section Docker.

Préparation et Configuration :

1. Téléchargement et Installation :

- Télécharger les fichiers Vagrant nécessaires depuis la section des ressources.
- Configurer un répertoire pour ces fichiers.
- S'assurer d'avoir 2 Go de mémoire vive et arrêter toutes les autres machines virtuelles pour éviter des conflits de ressources.

2. Lancement de la VM :

- Utiliser `vagrant up` pour démarrer la VM, installer Docker et Docker CLI.
- Se connecter à la VM via `vagrant ssh` et passer en utilisateur root.

Utilisation de Docker Compose :

1. Configuration de Docker Compose :

- Télécharger le fichier `docker-compose.yml` depuis le dépôt GitHub.
- Copier ce fichier dans la VM.

2. Lancer les Conteneurs :

- Utiliser `docker-compose up -d` pour démarrer les conteneurs en arrière-plan.
- Vérifier l'état des conteneurs avec `docker-compose ps`.

Validation de l'Application Vprofile :

• Accès et Test :

- Accéder aux conteneurs via l'adresse IP de la VM.
- Tester les différents services (Nginx, Tomcat, RabbitMQ, Memcached, MySQL) pour s'assurer qu'ils fonctionnent correctement.
- Utiliser l'interface utilisateur de Vprofile pour valider le bon fonctionnement de tous les conteneurs.

Nettoyage :

• Arrêt et Suppression :

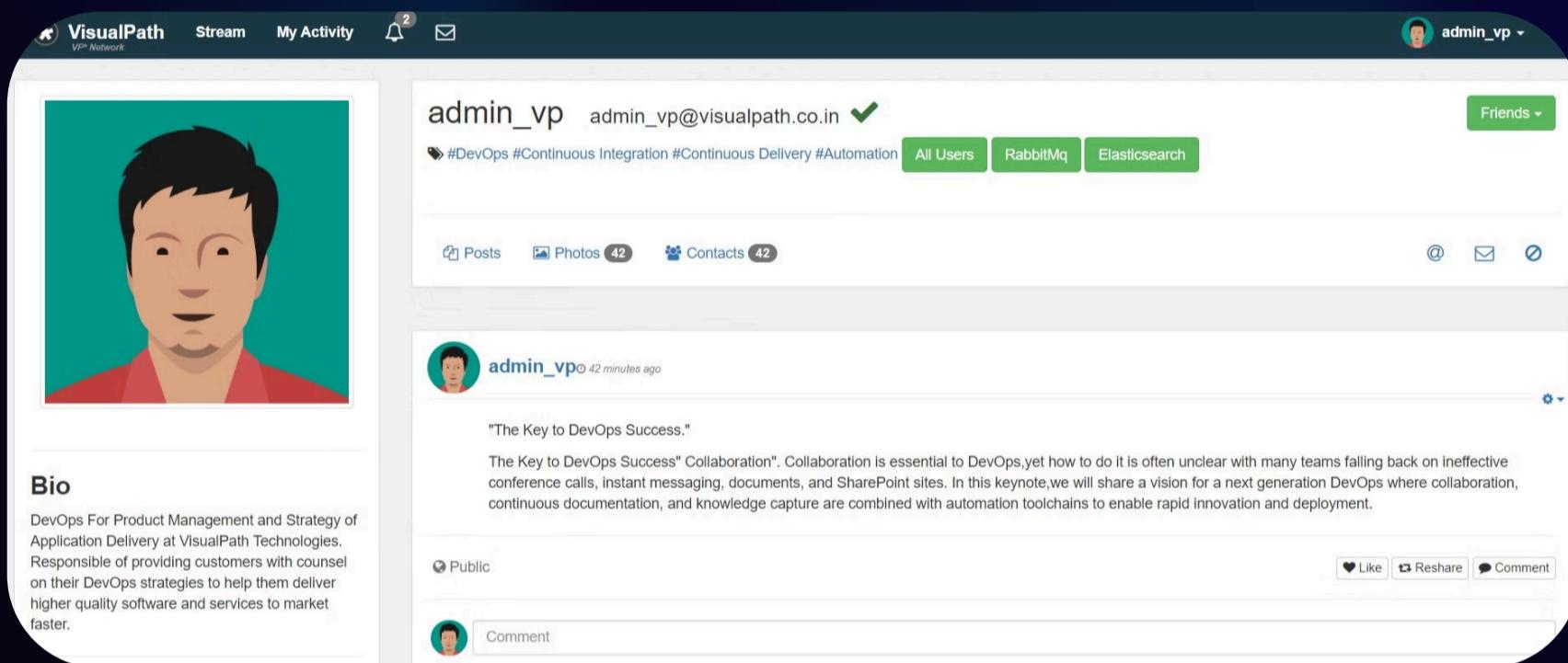
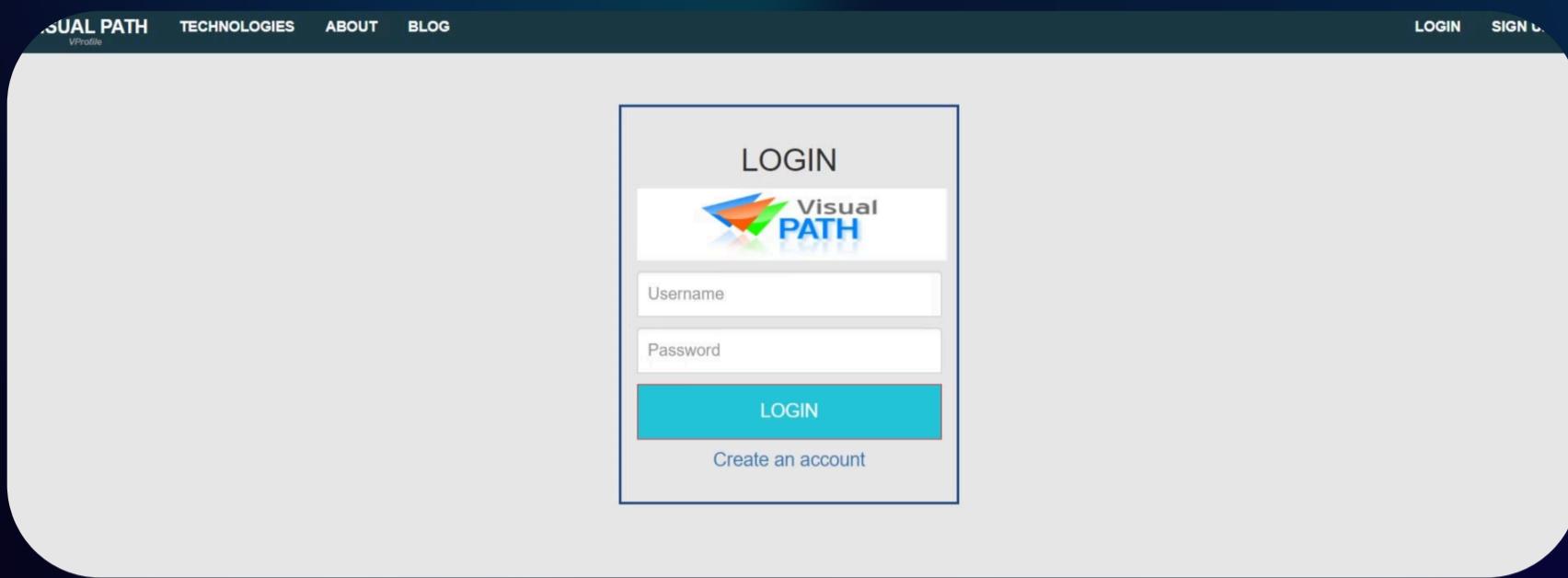
- Utiliser `docker-compose down` pour arrêter et supprimer les conteneurs.
- Utiliser `docker system prune -a` pour nettoyer toutes les images et conteneurs inutilisés.

Conclusion :

• Simplicité et Efficacité :

- Montrer la facilité d'exécution des applications sur des conteneurs Docker.
- Introduction à l'utilisation avancée de Docker, qui sera détaillée dans les sections suivantes de la conférence.

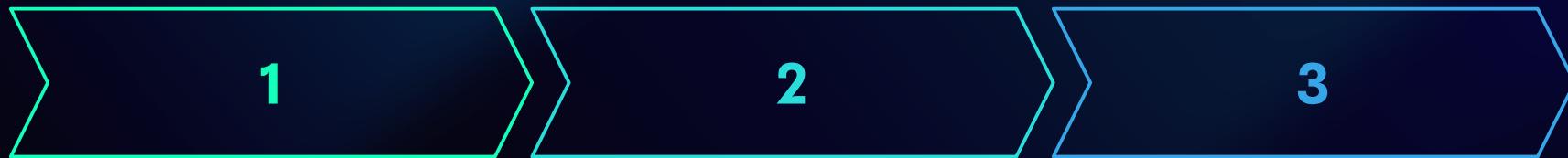
Projet Vprofile : Aller sur votre navigateur et entrer votre IP



User Name		User Id
admin_vp		7
WahidKhan		8
Gayatri		9
WahidKhan2		10
KiranKumar		11
Saikumar		12
RamSai		13

User Primary Details											
ID	Name	Father's Name	Mother's Name	Email	Phone Number	Date Of Birth	Gender	Marital Status	Permanent Address	Temporary Address	Primary Occupation
11	KiranKumar	KK	RK	kiran@gmail.com	1010101010	8/12/1993	male	unMarried	California	James Street	Software Engineer
User Extra Details											
Date Of Birth	Gender	Marital Status	Permanent Address	Temporary Address	Primary Occupation	Secondary Occupation	Skills	Secondary PhoneNumber	Nationality	Language	Working Experience
8/12/1993	male	unMarried	California	James Street	Software Engineer	Software Engineer	Java HTML CSS	1010101010	India	english	10

Les Microservices: Une Nouvelle Approche



Découpage Modulaire

Les microservices divisent les applications en composants indépendants et légers, facilitant le développement et la maintenance.

Communication Standardisée

Les microservices interagissent via des interfaces bien définies, assurant l'interopérabilité et la flexibilité.

Évolutivité Dynamique

Chaque microservice peut être déployé, mis à jour et redimensionné indépendamment, améliorant l'agilité.

Microservices en Pratique



Base de Données

Chaque microservice gère sa propre base de données, optimisée pour ses besoins spécifiques.



API Standardisées

Les microservices communiquent via des API RESTful bien définies, favorisant l'interopérabilité.



Déploiement Automatisé

L'intégration continue et le déploiement automatisé permettent une livraison rapide et fiable.



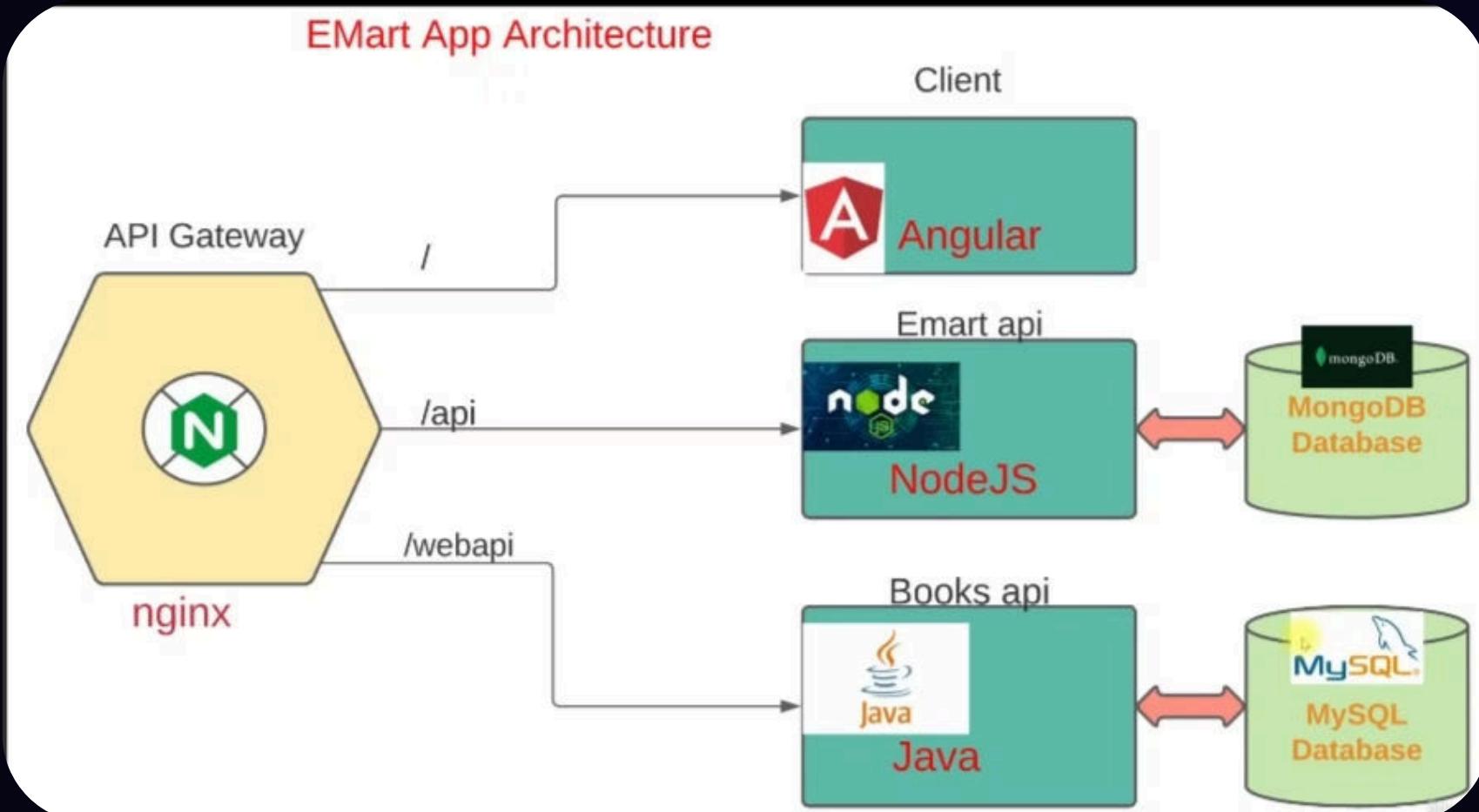
Évolutivité Élastique

Chaque microservice peut être redimensionné indépendamment en fonction de la charge de travail.

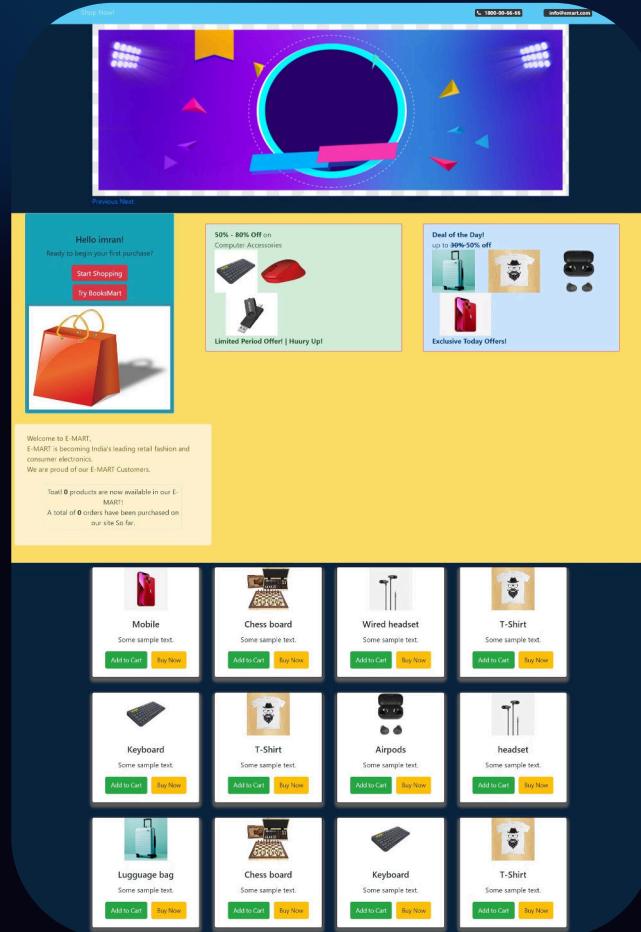


Découvrez les Microservices avec le Projet Vprofile

Frontend	Application web basée sur Spring Boot et Angular
Backend	Microservices implémentés avec Spring Boot et Java
Base de Données	MySQL, Redis et Elasticsearch pour le stockage des données
Orchestration	Kubernetes pour le déploiement et la gestion des conteneurs



Site Web Emart



Voici un résumé des points essentiels de la partie sur le déploiement de l'application microservices EMart :

Objectif et Contexte :

- **But** : Déployer l'application EMart, une application de commerce électronique, en utilisant une architecture de microservices.
- **Composants** :
 - **Frontend** : Passerelle API avec Nginx, application client en Angular.
 - **Backend** : Services NodeJS pour l'API principale, Java pour l'API des livres, MongoDB pour la base de données principale, et MySQL pour la base de données des livres.

Configuration et Déploiement :

1. Préparation de l'Environnement :

- Télécharger les fichiers nécessaires, configurer le répertoire et s'assurer que toutes les autres machines virtuelles sont arrêtées.
- Lancer la VM avec `vagrant up` et se connecter avec `vagrant ssh`.

2. Utilisation de Docker Compose :

- Cloner le dépôt de l'application EMart.
- Le fichier `docker-compose.yml` inclut une étape de build pour créer des images Docker à partir du code source.
- Lancer les conteneurs avec `docker-compose up -d`.

Validation de l'Application EMart :

• Accès et Test :

- Accéder à l'application via l'adresse IP de la VM.
- Tester les fonctionnalités : enregistrement et connexion d'utilisateur, interaction avec les services backend (NodeJS et Java), et vérification de la communication avec les bases de données (MongoDB et MySQL).

Nettoyage :

• Arrêt et Suppression :

- Utiliser `docker-compose down` pour arrêter et supprimer les conteneurs.
- Utiliser `docker system prune -a` pour nettoyer toutes les images et conteneurs inutilisés.
- Quitter la VM et s'assurer qu'elle est arrêtée avec `vagrant halt`.

Conclusion :

• Simplicité et Scalabilité :

- Montrer la facilité d'exécution d'une application microservices avec Docker et Docker Compose.
- Introduction à l'utilisation avancée de Docker, avec des exemples concrets de build et de gestion de conteneurs.

Docker Images, Docker Containers et Docker Compose : Différences et Commandes Essentielles

Docker Images

Concept : Une image Docker est un package léger et autonome qui inclut tout ce qui est nécessaire pour exécuter un morceau de logiciel, y compris le code, un runtime, des bibliothèques, des variables d'environnement et des configurations. Les images servent de modèles pour créer des conteneurs.

Commandes Essentielles :

1. `docker images` : Liste les images Docker locales.
2. `docker pull <image>` : Télécharge une image depuis un registre Docker (comme Docker Hub).
3. `docker build -t <nom_image> .` : Construit une image Docker à partir d'un Dockerfile situé dans le répertoire courant.
4. `docker rmi <image_id>` : Supprime une image Docker locale.

Docker Containers

Concept : Un conteneur Docker est une instance en cours d'exécution d'une image Docker. Les conteneurs sont légers et isolés, mais partagent le noyau du système d'exploitation de l'hôte.

Commandes Essentielles :

1. `docker ps` : Liste les conteneurs Docker en cours d'exécution.
2. `docker ps -a` : Liste tous les conteneurs, y compris ceux qui sont arrêtés.
3. `docker run -d -p <port_hote>:<port_conteneur> --name <nom_conteneur> <image>` : Crée et exécute un nouveau conteneur à partir d'une image.
4. `docker stop <nom_conteneur>` : Arrête un conteneur en cours d'exécution.
5. `docker rm <nom_conteneur>` : Supprime un conteneur arrêté.
6. `docker logs <nom_conteneur>` : Affiche les logs d'un conteneur.
7. `docker exec -it <nom_conteneur> /bin/bash` : Ouvre un terminal interactif dans un conteneur en cours d'exécution.

Docker Compose

Concept : Docker Compose est un outil qui permet de définir et de gérer des applications multi-conteneurs. Avec un fichier `docker-compose.yml`, vous pouvez configurer les services, les réseaux et les volumes de votre application.

Commandes Essentielles :

1. `docker-compose up -d` : Démarré tous les services définis dans le fichier `docker-compose.yml` en arrière-plan.
2. `docker-compose down` : Arrête et supprime tous les conteneurs, réseaux et volumes définis par `docker-compose up`.
3. `docker-compose build` : Construit ou reconstruit les images de service.
4. `docker-compose logs` : Affiche les logs de tous les services.
5. `docker-compose ps` : Liste les conteneurs gérés par Docker Compose.
6. `docker-compose exec <service> <commande>` : Exécute une commande dans un conteneur en cours d'exécution.