

Dernier brief Docker & Jenkins

Jenkins :

Jenkins : Jenkins est un outil open-source d'intégration continue et de déploiement continu largement utilisé. Il permet d'automatiser les processus de construction, de test et de déploiement des logiciels.

Pipeline Jenkins : Une pipeline Jenkins est une suite d'étapes qui permettent de définir, de contrôler et d'automatiser le flux de travail de développement logiciel, de la construction à la livraison. Elle est généralement définie dans un fichier texte, tel qu'un Jenkins File, et peut être exécutée de manière répétable.

Agent Jenkins : Un agent Jenkins est un composant qui exécute les tâches de la pipeline Jenkins. Il peut s'agir d'une machine physique ou virtuelle, ou même d'un conteneur Docker. Les agents Jenkins communiquent avec le serveur Jenkins pour recevoir des instructions sur les tâches à exécuter.

Job Jenkins : Un job Jenkins est une tâche spécifique à accomplir, comme la compilation d'un projet, l'exécution de tests, la construction d'un conteneur Docker, etc. Les jobs sont configurés dans Jenkins pour exécuter différentes étapes du pipeline.

Stage : Un stage dans une pipeline Jenkins représente une étape spécifique du processus de construction et de déploiement. Chaque stage peut avoir plusieurs étapes ou actions à exécuter, comme la compilation du code source, l'exécution des tests, la construction d'une image Docker, etc.

Script Pipeline : Le script pipeline, souvent écrit en Groovy, est une manière de définir la logique et le comportement de chaque étape et action dans la pipeline Jenkins. Il permet de personnaliser le flux de travail en fonction des besoins spécifiques du projet.

Plugin Jenkins : Les plugins Jenkins sont des modules additionnels qui étendent les fonctionnalités de base de Jenkins. Ils peuvent être utilisés pour intégrer Jenkins avec d'autres outils et services, ajouter de nouvelles fonctionnalités, ou fournir des intégrations spécifiques pour différents types de projets et technologies.

Intégration continue (CI) : L'intégration continue est une pratique de développement logiciel qui consiste à automatiser le processus de construction et de test du code à chaque modification apportée au référentiel. L'objectif est d'identifier

rapidement les erreurs et de s'assurer que le code nouvellement développé fonctionne correctement avec le reste du système.

Déploiement continu (CD) : Le déploiement continu est une extension de l'intégration continue qui consiste à automatiser le processus de déploiement du code en production dès qu'il a été testé et validé. L'objectif est d'accélérer la livraison des nouvelles fonctionnalités aux utilisateurs finaux tout en maintenant la qualité du logiciel.

En utilisant ces concepts et en configurant Jenkins en conséquence, les équipes de développement peuvent automatiser efficacement leurs processus de développement logiciel, améliorer la qualité du code et accélérer le déploiement des applications.

Docker :

Docker : Docker est une plateforme open-source qui permet de créer, déployer et exécuter des applications dans des conteneurs logiciels. Les conteneurs Docker sont des environnements légers et portables qui contiennent tout ce dont une application a besoin pour s'exécuter, y compris le code, les bibliothèques, les dépendances et les variables d'environnement.

Conteneur Docker : Un conteneur Docker est une instance d'une image Docker en cours d'exécution. Il s'agit d'une unité d'isolation qui encapsule une application et toutes ses dépendances, tout en partageant le noyau du système d'exploitation de l'hôte. Les conteneurs Docker sont légers, rapides à démarrer et à arrêter, et peuvent fonctionner de manière cohérente dans différents environnements.

Image Docker : Une image Docker est un modèle de conteneur réutilisable qui contient tous les éléments nécessaires pour exécuter une application, y compris le système d'exploitation, les bibliothèques, les dépendances et le code source. Les images Docker sont créées à partir de fichiers Dockerfile, qui spécifient les étapes pour construire l'image.

Dockerfile : Un Dockerfile est un fichier texte qui contient les instructions pour construire une image Docker. Il définit les étapes nécessaires pour configurer l'environnement d'exécution de l'application, installer les dépendances, copier le code source, et configurer les paramètres de l'application.

Registry Docker : Un registre Docker est un dépôt centralisé qui stocke et distribue des images Docker. Le registre Docker par défaut est Docker Hub, mais il existe

d'autres registries publics et privés disponibles, tels que Amazon ECR, Google Container Registry, et Azure Container Registry.

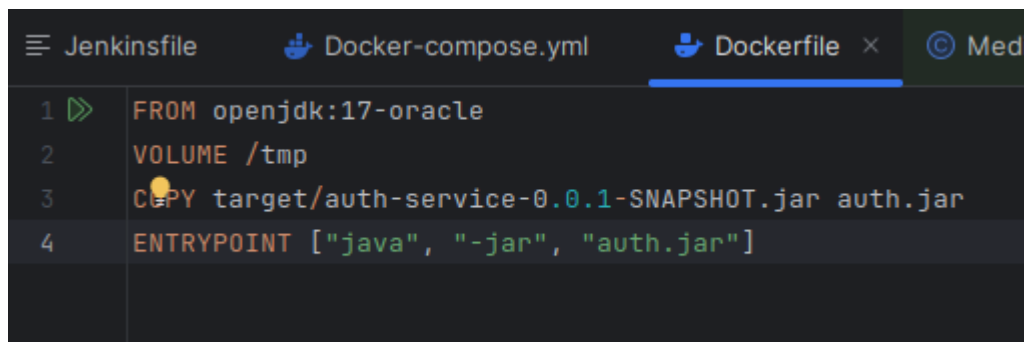
Docker Compose : Docker Compose est un outil qui permet de définir et de gérer des applications multi-conteneurs à l'aide d'un fichier YAML appelé `docker-compose.yml`. Il simplifie le processus de configuration et de déploiement d'applications composées de plusieurs services interconnectés.

Docker Swarm : Docker Swarm est un outil de clustering et d'orchestration intégré à Docker qui permet de gérer un groupe de conteneurs Docker répartis sur plusieurs hôtes. Il offre des fonctionnalités telles que le déploiement automatique, l'équilibrage de charge, la mise à l'échelle automatique et la haute disponibilité des applications.

Kubernetes (K8s) : Kubernetes est une plateforme open-source d'orchestration de conteneurs qui permet de déployer, gérer et scaler des applications conteneurisées dans un environnement cloud ou sur site. Il offre des fonctionnalités avancées pour la gestion des conteneurs, telles que l'auto-scaling, la gestion des ressources, la gestion des volumes et la tolérance aux pannes.

En utilisant ces concepts et outils, Docker permet aux développeurs et aux équipes d'opérations de créer des environnements d'exécution isolés, portables et reproductibles pour leurs applications, ce qui facilite le développement, le déploiement et la gestion des applications modernes.

- Création de Dockerfile pour chaque service pour builder images d'un chaque service:



```
Jenkinsfile  Docker-compose.yml  Dockerfile x  Medi
1  FROM openjdk:17-oracle
2  VOLUME /tmp
3  COPY target/auth-service-0.0.1-SNAPSHOT.jar auth.jar
4  ENTRYPOINT ["java", "-jar", "auth.jar"]
```

- Création de docker-compose file regrouper tous les images des services d'application et créer un network commun :

```
Jenkinsfile  Docker-compose.yml x  Dockerfile  MediaServiceApplicationTests.java
19
20 social-network-discovery-service:
21   build: ./discovery
22   container_name: social-network-discovery-service
23   ports:
24     - '8761:8761'
25   expose:
26     - '8761'
27   healthcheck:
28     test: ["CMD", "curl", "-f", "http://localhost:8761/actuator/health"]
29     interval: 10s
30   networks:
31     - social-network-app
32
33 social-network-auth-service:
34   build: ./auth-service
35   container_name: social-network-auth-service
36   ports:
37     - '8101:8101'
38   expose:
39     - '8101'
40   networks:
41     - social-network-app
42   depends_on:
43     - social-network-geteway-service
44
45 social-network-geteway-service:
```

- Création de Jenkins File pour exécuter les stages:

```

7      git 'git'
8    }
9
10   stages {
11     stage('Checkout') {
12       steps {
13         git branch: 'main', url: 'https://github.com/Mouslih1/social-network-microservice'
14       }
15     }
16
17     stage('Build') {
18       steps {
19         script {
20           def microservices = ['discovery','getaway', 'media-service','auth-service', 'Frie
21
22           microservices.each { service ->
23             dir(service) {
24               if (isUnix()) {
25                 sh 'mvn clean install'
26               } else {
27                 bat 'mvn clean install'
28               }
29             }
30           }
31         }
32       }
33     }
34   }

```

- Après configurer le pipeline avec c'est tools :
- Voici le résultat de pipeline et les stages exécuter:

