

Rapport Projet : Architecture des composants d'entreprise

Imane Chaourad – Oussama Rida – Hatim Ouled Moussa

I -Introduction :

À l'ère du numérique, la création de blogs personnels a gagné en popularité en tant que moyen puissant de partager des idées, des expériences et des connaissances avec un large public. Dans ce contexte, le développement d'un blog personnel doté de fonctionnalités avancées revêt une importance particulière. Ce projet vise à fournir une plateforme dynamique permettant à l'utilisateur de publier des articles, d'interagir avec son audience via des commentaires, et de faciliter la recherche pour une expérience utilisateur enrichissante.

Aperçu du Projet :

Le projet consiste en la création d'un blog personnel doté de fonctionnalités robustes, centrées sur la publication, l'interaction et la recherche. Les principales caractéristiques incluent :

Publication d'Articles : L'utilisateur aura la possibilité de créer, éditer et publier des articles de manière intuitive. Une interface conviviale permettra d'ajouter du contenu textuel, des médias et de personnaliser la présentation visuelle.

Système de Commentaires : Les lecteurs pourront interagir avec les articles en laissant des commentaires, favorisant ainsi un échange d'idées et de perspectives. Des fonctionnalités de modération garantiront une expérience communautaire positive.

Fonctionnalités de Recherche : Un moteur de recherche puissant sera intégré pour permettre aux visiteurs de trouver rapidement des articles pertinents en fonction de mots-clés, de catégories ou de tags.

Gestion des Utilisateurs : Une interface d'administration permettra à l'utilisateur de gérer son contenu, de suivre les commentaires et d'effectuer des ajustements personnalisés.

Importance de l'Architecture Microservices :

L'architecture microservices s'avère cruciale dans ce projet pour plusieurs raisons. Elle offre une approche modulaire, permettant le développement, le déploiement et la mise à l'échelle indépendante de chaque service. Les avantages incluent :

Scalabilité : Les microservices permettent de faire évoluer individuellement les composants nécessaires en fonction des besoins, garantissant une extensibilité efficace du blog.

Flexibilité Technologique : Chaque service peut être développé avec la technologie la mieux adaptée à sa fonction spécifique, optimisant ainsi les performances et la maintenance.

Facilité de Déploiement : La mise en place de microservices simplifie le déploiement continu, permettant des mises à jour fréquentes sans perturber l'ensemble du système.

Résilience et Tolérance aux Pannes : L'indépendance des microservices favorise une meilleure résilience globale, limitant l'impact d'éventuelles défaillances à des services spécifiques.

II- Architecture Microservice

Architecture

L'architecture du projet de blog personnel utilisant une approche microservices est basée sur une décomposition modulaire de l'application en plusieurs services indépendants. Chaque service se concentre sur une fonctionnalité spécifique, ce qui permet une évolution, un déploiement et une maintenance indépendants.

Voici une vue d'ensemble de l'architecture du projet :

Service de Commentaires :

Fonctionnalité : Gère les commentaires associés aux articles du blog.

Technologies : API RESTful pour les opérations CRUD, base de données MySQL pour stocker les commentaires.

Service Utilisateur :

Fonctionnalité : Gère les informations relatives aux utilisateurs, y compris l'authentification.

Technologies : API RESTful pour la gestion des utilisateurs, stockage des données utilisateur dans une base de données sécurisée.

Service de Blog :

Fonctionnalité : Gère la création, l'édition et la suppression d'articles de blog.

Technologies : API RESTful pour la gestion des articles, base de données MySQL pour stocker les articles.

Service de Likes :

Fonctionnalité : Gère les interactions "J'aime" des utilisateurs sur les articles.

Technologies : API RESTful pour les opérations de like, base de données pour enregistrer les relations entre utilisateurs et articles.

Service de Gateway (Passerelle) :

Fonctionnalité : Point d'entrée unique pour le système, agréant les différents endpoints des microservices.

Technologies : API Gateway Netflix , gestion de l'authentification

Service Eureka (Service de Registre) :

Fonctionnalité : Gère la découverte des services enregistrés dans l'écosystème.

Technologies : Service de registre Netflix Eureka pour la découverte dynamique des services.

Description des services :

1 -Service de Commentaires :

Fonctionnalité : Gère la création, la modification, la suppression et la consultation des commentaires associés aux articles du blog.

API : Expose des endpoints pour ajouter des commentaires, récupérer tous les commentaires d'un article, supprimer un commentaire, etc.

Base de Données : Stocke les informations relatives aux commentaires, y compris le contenu, l'auteur, la date et l'article associé.

2 -Service Utilisateur :

Fonctionnalité : Gère les informations relatives aux utilisateurs, y compris l'authentification, la création de compte et la gestion du profil.

API : Permet l'inscription, la connexion, la récupération des détails de profil, la mise à jour des informations de compte, etc.

Base de Données : Stocke les données des utilisateurs, telles que les noms, les adresses e-mail, les mots de passe (de manière sécurisée), et d'autres informations de profil.

3-Service de Blog :

Fonctionnalité : Gère la création, l'édition et la suppression d'articles de blog.

API : Permet la publication d'articles, la récupération de la liste des articles, la mise à jour d'articles existants, etc.

Base de Données : Stocke les articles du blog, y compris le contenu, l'auteur, la date de publication et les informations associées.

4-Service de Likes :

Fonctionnalité : Gère les interactions "J'aime" des utilisateurs sur les articles.

API : Fournit des endpoints pour ajouter ou retirer un "J'aime" à un article, récupérer le nombre total de "J'aime", etc.

Base de Données : Enregistre les relations entre les utilisateurs et les articles qu'ils ont aimés.

5-Service de Gateway (Passerelle) :

Fonctionnalité : Fonctionne comme un point d'entrée unique pour le système, dirige les requêtes vers les microservices appropriés.

API : Agrège les différents endpoints des microservices et fournit une interface unifiée pour les clients.

Fonctionnalités avancées : Gestion de l'authentification, de l'autorisation, équilibrage de charge, routage intelligent.

6-Service Eureka (Service de Registre) :

Fonctionnalité : Gère la découverte de services enregistrés dans l'écosystème, permettant aux services de trouver et de communiquer entre eux dynamiquement.

API : Fournit une interface pour l'inscription et la récupération des informations sur les services.

Fonctionnalités avancées : Garantit la résilience en mettant à jour automatiquement la liste des services disponibles et en retirant ceux qui ne répondent pas.

III- Conception des microservices

Service de Commentaires :

Approche : Adopte une approche centrée sur la gestion des commentaires associés aux articles du blog.

Responsabilités : Gérer la création, la modification, la suppression et la consultation des commentaires. Assurer la cohérence des données liées aux commentaires.

Service Utilisateur :

Approche : Axé sur la gestion des utilisateurs et de leurs profils dans le système.

Responsabilités : Gérer l'authentification, la création de compte, la gestion des profils utilisateur et la sécurité des données utilisateur.

Service de Blog :

Approche : Concentré sur la publication et la gestion des articles de blog.

Responsabilités : Gérer la création, la mise à jour et la suppression d'articles. Fournir des fonctionnalités pour récupérer la liste des articles et leurs détails.

Service de Likes :

Approche : Se concentre sur la gestion des interactions "J'aime" entre les utilisateurs et les articles.

Responsabilités : Gérer les actions de "J'aime" et de "Je n'aime pas". Fournir des informations sur le nombre total de "J'aime" pour un article donné.

Service de Gateway (Passerelle) :

Approche : Fournit un point d'entrée unique pour le système, facilitant l'accès aux fonctionnalités des microservices sous-jacents.

Responsabilités : Agréger les différents endpoints des microservices, gérer l'authentification, l'autorisation, l'équilibrage de charge et offrir une interface unifiée pour les clients.

Service Eureka (Service de Registre) :

Approche : Axé sur la gestion de la découverte des services au sein de l'écosystème microservices.

Responsabilités : Enregistrer les informations sur les services disponibles, faciliter la découverte dynamique des services, garantir la résilience en actualisant automatiquement la liste des services.

IV- Conteneurisation avec Docker

La conteneurisation avec Docker est une technologie populaire qui simplifie le déploiement, la gestion et la mise à l'échelle d'applications.

Voici une explication de l'implémentation de Docker et de ses avantages :

Avantages de la conteneurisation avec Docker :

- **Portabilité :** Les conteneurs Docker encapsulent l'application et ses dépendances, assurant ainsi la portabilité entre les environnements de développement, de test et de production. Vous pouvez exécuter le même conteneur sur n'importe quel système compatible Docker.
- **Isolation :** Les conteneurs fournissent une isolation légère, garantissant que les applications ne perturbent pas l'environnement les uns des autres. Chaque conteneur a son propre espace de noms et ses ressources, ce qui évite les conflits.
- **Rapidité et efficacité :** Les conteneurs partagent le noyau du système hôte, ce qui les rend plus légers que les machines virtuelles. Cela permet un démarrage rapide des conteneurs et une utilisation plus efficace des ressources système.
- **Facilité de déploiement :** Docker simplifie le processus de déploiement en automatisant la gestion des dépendances et en fournissant une configuration reproductible. Cela facilite la mise à l'échelle horizontale et verticale des applications.
- **Gestion des versions :** Les images Docker peuvent être versionnées, ce qui facilite le suivi des changements et le retour en arrière en cas de besoin.
- **Écosystème riche :** Docker s'accompagne d'un écosystème riche de registres publics (comme Docker Hub) où vous pouvez partager vos images,

Implémentation de la conteneurisation avec Docker :

On commence par la création des Dockerfiles dans la racine de chaque microservice

 .mvn/wrapper	first commit	2 days ago
 .scannerwork	first commit	2 days ago
 src	first commit	2 days ago
 .gitignore	first commit	2 days ago
 Dockerfile	first commit	2 days ago
 Jenkinsfile	first com	2 days ago
 mvnw	first commit	2 days ago
 mvnw.cmd	first commit	2 days ago
 pom.xml	first com	2 days ago
 sonar-project.properties	first comi	2 days ago

Le script est comme ceci :

```
Code Blame 7 lines (4 loc) · 128 Bytes Code 55% faster with GitHub Copilot

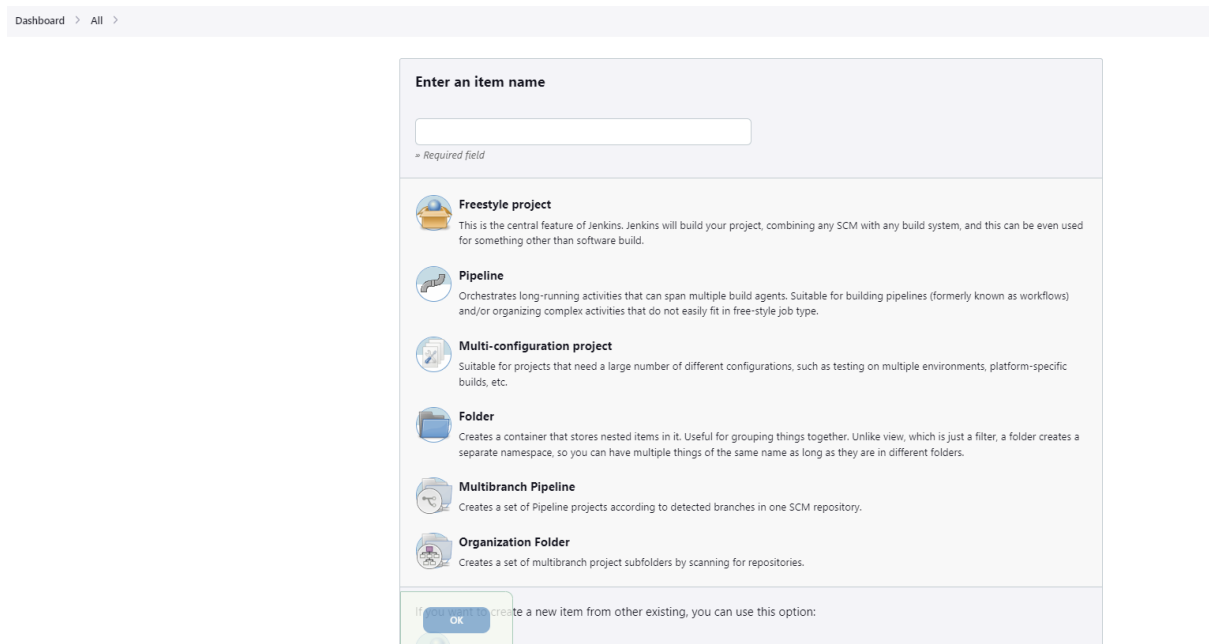
1 FROM openjdk:17
2 WORKDIR /App
3
4 COPY /target/blogs-0.0.1-SNAPSHOT.jar .
5
6
7 ENTRYPOINT ["java", "-jar" , "blogs-0.0.1-SNAPSHOT.jar"]
```

V- CI/CD avec Jenkins & Déploiement Automatique

Jenkins est un outil d'intégration continue (CI) et de déploiement continu (CD) populaire, permettant d'automatiser les processus de construction, de test et de déploiement. Voici comment vous pouvez mettre en place un pipeline CI/CD avec Jenkins

Processus

Après l'installation et configuration de Jenkins : On créer un nouveau projet pipelines



On configure le projet

Configure

General

Advanced Project Options

Pipeline

☐ Do not allow the pipeline to resume if the controller restarts

☒ GitHub project

Project url ?

Advanced ▾

☐ Pipeline speed/durability override ?

☐ Preserve stashes from completed builds ?

☐ This project is parameterized ?

☐ Throttle builds ?

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☐ GitHub Branches

☐ GitHub Pull Requests ?

☒ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

☐ Quiet period ?

☐ Trigger builds remotely (e.g., from scripts) ?

Save

Apply

Et puis on créer le script « le pipeline » ou on définissent l'agent les stages et les steps de l'automatisation : qui est Clone >> Build >> Create Docker Image >> Run

Configure

General

Advanced Project Options

Pipeline

Pipeline

Definition

Pipeline script ▾

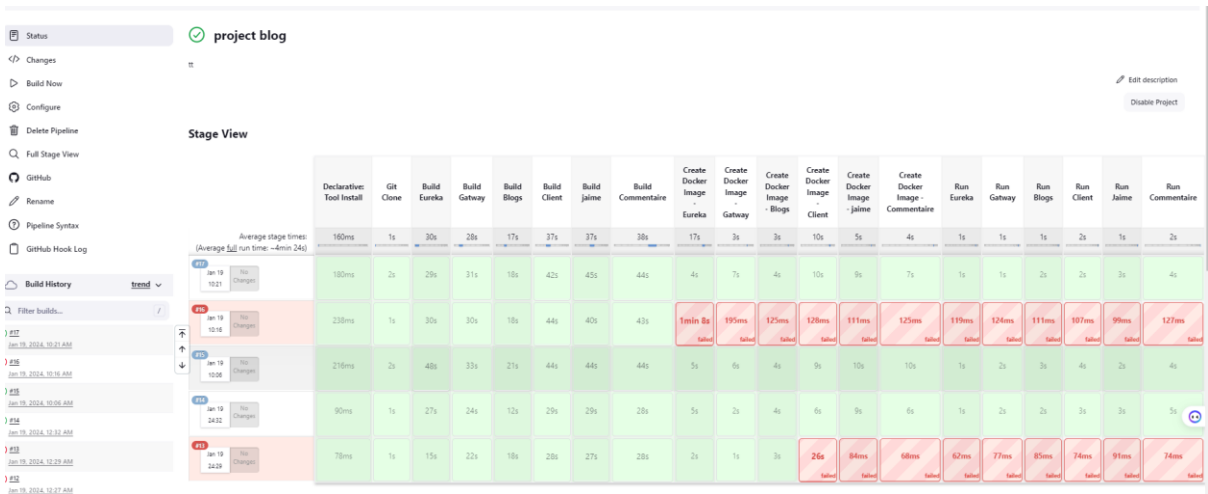
Script ?

```
1= pipeline {
2=   agent any
3=   tools {
4=     maven 'maven'
5=   }
6=
7=   stages {
8=     stage('git Clone') {
9=       steps {
10=        script {
11=          checkout([class: 'GitSCM', branches: [[name: 'master']], userRemoteConfigs: [[url: 'https://github.com/hatim9681/personal-blog.git']]])
12=        }
13=      }
14=    }
15=
16=    stage('Build Eureka') {
17=      steps {
18=        dir('C:\\Users\\Dell\\Desktop\\Blog Personnel\\Blog-backend\\Eureka_server') {
19=          bat 'mvn clean install'
20=        }
21=      }
22=    }
23=
24=    stage('Build gateway') {
25=      steps {
26=        dir('C:\\Users\\Dell\\Desktop\\Blog Personnel\\Blog-backend\\gateway') {
27=          bat 'mvn clean install'
28=        }
29=      }
30=    }
31=
32=    stage('Build Blogs') {
33=      steps {
34=        dir('C:\\Users\\Dell\\Desktop\\Blog Personnel\\Blog-backend\\blogs') {
35=          bat 'mvn clean install'
36=        }
37=      }
38=    }
39=
40=    stage('Build Client') {
41=      steps {
42=        dir('C:\\Users\\Dell\\Desktop\\Blog Personnel\\Blog-backend\\client') {
```

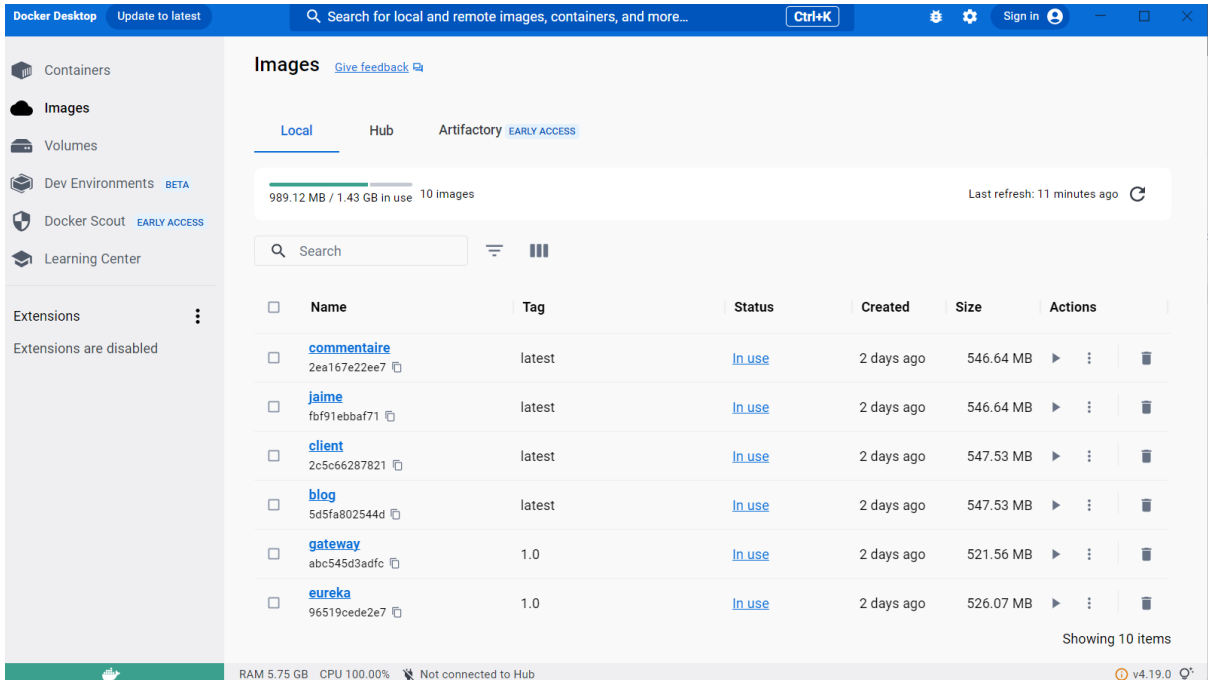
Save

Apply

Et puis on « build now » le projet



Les images ont été bien créer



On leur exécution a générer des containers qui se sont bien lancé

Docker Desktop

Update to latest

Search for local and remote images, containers, and more...

Ctrl+K

Sign in

Containers

Images

Volumes

Dev Environments BETA

Docker Scout EARLY ACCESS

Learning Center

Extensions

































Extensions are disabled

Containers

Give feedback

Search

Only show running containers

	Name	Image	Status	Port(s)	Last started	Actions
<input type="checkbox"/>	 commentaire-poss cf292b5abc36	commentaire	Running	8088:8088	9 seconds ago	  
<input type="checkbox"/>	 jaime-poss e6cad2a2497d	jaime	Running	8086:8086	8 seconds ago	  
<input type="checkbox"/>	 client-poss 467ed88eecf0	client	Running	8084:8084	8 seconds ago	  
<input type="checkbox"/>	 blog-poss 4681cab36b79	blog	Running	8081:8081	9 seconds ago	  
<input type="checkbox"/>	 gateway-poss 18d01206a9ee	gateway:1.0	Running	8089:8089	9 seconds ago	  
<input type="checkbox"/>	 eureka-poss 86609097e936	eureka:1.0	Running	8761:8761	9 seconds ago	  
<input type="checkbox"/>	 rabbit-server a7bb9fa8bf60	rabbitmq:3.12.9-management	Exited (255)	15672:15672 Show all ports (2)	2 days ago	  
<input type="checkbox"/>	 unruffled_babbage 5d4627aaa2b5	rabbitmq	Exited		9 days ago	  

Showing 9 items

RAM 5.31 GB CPU 100.00% Not connected to Hub v4.19.0

VI- Intégration de SonarQube

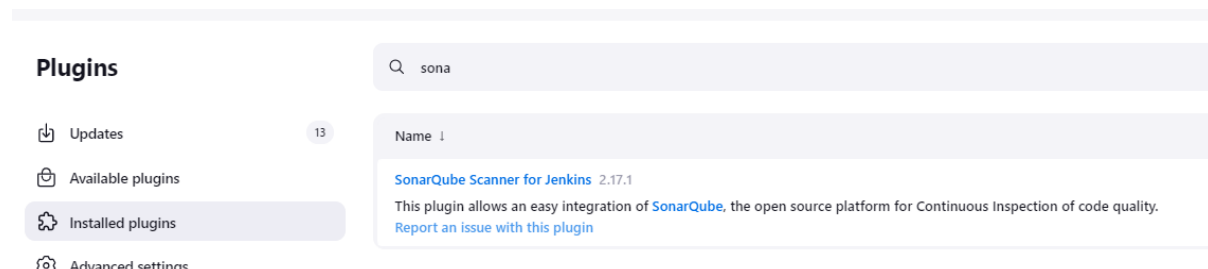
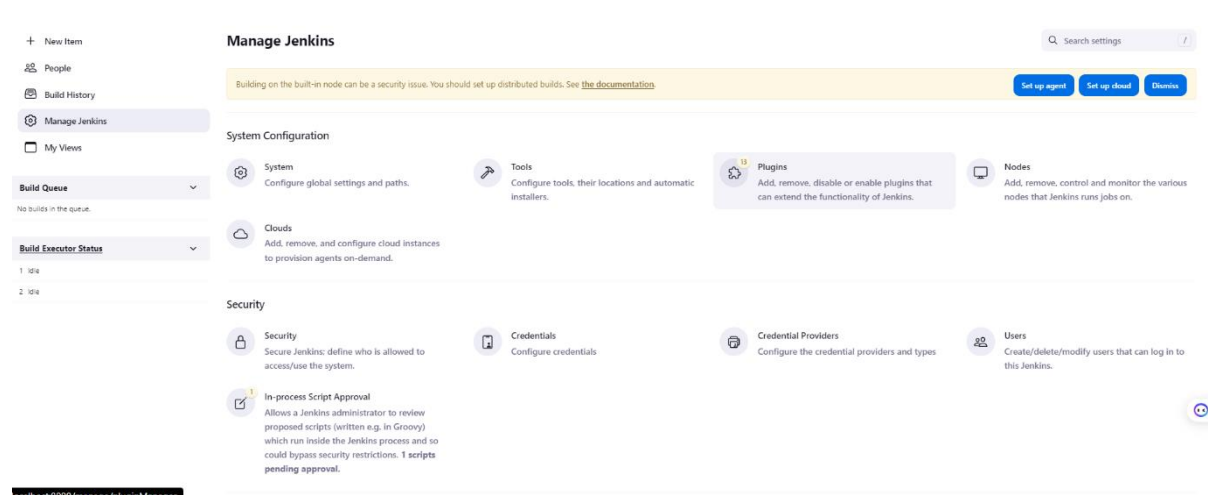
SonarQube est une plateforme open-source d'analyse statique de code qui permet d'assurer la qualité du code source en détectant les vulnérabilités, les bugs, les erreurs de conception et autres violations des bonnes pratiques de programmation :

Voici quatre bénéfices succincts de l'intégration de SonarQube dans un pipeline CI/CD :

- **Détection précoce des problèmes** : Identifie rapidement les vulnérabilités, bugs et violations de bonnes pratiques de codage.
- **Amélioration de la cohérence du code** : Favorise la cohérence en signalant les écarts par rapport aux normes de codage.
- **Gestion de la dette technique** : Évalue et aide à gérer l'accumulation de dette technique au fil du temps.
- **Intégration dans la revue de code** : Utilise les analyses comme référence lors des revues de code, renforçant les standards de qualité.

Voici comment intégrer SonarQube dans un pipeline CI/CD :

Installation du plugin



On Configure le sonarqube serveur , pour mon cas le url c'est adresse de sonar cloud vu que j'ai utilisé le serveur sonarqube du cloud

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☒ Environment variables

SonarQube installations

List of SonarQube installations

Name

blog

Server URL

Default is http://localhost:9000

https://sonarcloud.io

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

- none -

+ Add

Advanced

Add SonarQube

Save Apply

Puis on ajoute l'authentification token généré par sonarqube

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Global credentials (unrestricted)

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?











client

☐ Treat username as secret ?

Password ?

testsona

On définit le path du script en étant un jenkins file créer dans la racine du projet

 .mvn/wrapper	first commit	2 days ago
 .scannerwork	first commit	2 days ago
 src	first commit	2 days ago
 .gitignore	first commit	2 days ago
 Dockerfile	first commit	2 days ago
 Jenkinsfile	first com	2 days ago
 mvnw	first commit	2 days ago
 mvnw.cmd	first commit	2 days ago
 pom.xml	first com	2 days ago
 sonar-project.properties	first comi	2 days ago

Sa configuration

Code

Blame

19 lines (17 loc) · 378 Bytes

 Code 55% faster with GitHub Copilot

```
1  pipeline {
2      agent any
3
4      options {
5          buildDiscarder(logRotator(numToKeepStr: '5'))
6      }
7
8      stages {
9          stage('SonarQube Analysis') {
10              steps {
11                  script {
12                      withSonarQubeEnv(installationName: 'blog') {
13                          bat 'sonar-scanner'
14                      }
15                  }
16              }
17          }
18      }
19  }
```

Une fois terminer, on exécute notre pipeline

Status

</> Changes

► Build Now

⚙️ Configure

🗑️ Delete Pipeline

🔍 Full Stage View

🌊 SonarQube

✎ Rename

❓ Pipeline Syntax

Build History

trend ▾

Filter builds...

#12 Jan 19, 2024, 2:22 AM

#11 Jan 19, 2024, 2:16 AM

#10 Jan 19, 2024, 2:13 AM

#9 Jan 19, 2024, 2:12 AM

#8 Jan 19, 2024, 2:12 AM

Atom feed for all

Atom feed for failures

blog

Stage View

Average stage times:
(Average full run time: ~30s)

	Declarative: Checkout SCM	SonarQube Analysis
#12 Jan 19 02:22 1 commit	1s	27s
#11 Jan 19 02:16 1 commit	1s	12s failed
#10 Jan 19 02:13 No Changes	1s	26s failed
#9 Jan 19 02:12 No Changes	1s	6s failed
#8 Jan 19 02:12 No Changes	1s	5s failed

Un nouveau bar item s'ajoute « SonarQube », clique dessus et ca nous montre notre rapport

sonarcloud

client

Overview

Main Branch

Pull Requests

Branches

Information

Administration

Collapse

My Projects

My Issues

Explore

Q

client > client > Overview

client

No tags

Last analysis Dec 31, 2023 603 Lines of Code

Main Branch Status

Quality Gate Passed

Main Branch Evolution since 20 days ago

56 Findings

Bugs

Vulnerabilities

Code Smells

Security Hotspots

IV- Conclusion :

En conclusion, la création d'un blog personnel doté de fonctionnalités avancées représente une opportunité passionnante d'explorer et de partager des idées dans le monde numérique. Le projet met en lumière l'importance d'une architecture microservices pour garantir la scalabilité, la flexibilité et la résilience du système. Chaque microservice, avec son propre ensemble de responsabilités, contribue à créer un écosystème modulaire et hautement performant.

En adoptant une approche centrée sur chaque service, le système assure une gestion efficace des commentaires, des utilisateurs, des articles de blog et des interactions "J'aime". La passerelle joue un rôle crucial en fournissant un point d'entrée unifié, facilitant l'accès aux fonctionnalités tout en gérant des aspects tels que l'authentification et l'autorisation. De même, le service Eureka simplifie la découverte des services, contribuant ainsi à la résilience du système.

Cette conception microservices permet une évolution indépendante des composants, une facilité de déploiement et une meilleure gestion des pannes. En somme, le projet aspire à offrir une expérience utilisateur enrichissante tout en relevant les défis techniques avec une architecture moderne et adaptable.