

Démos : Etats des lieux

Table des matières

Copilot.....	2
TDD.....	2
MergeRequest Gitlab.....	2
Mise en place.....	2
Milestones, Issues, Labels, Tableaux de bord.....	2
MergeRequest et GitlabFlow.....	3
CI/CD.....	4
Analyses Qualité / Vulnérabilités.....	5
Analyse statique.....	5
Analyse dynamique.....	5
Déploiement immuable.....	5
Observabilité.....	5

Copilot

<https://docs.github.com/fr/copilot/using-github-copilot/getting-started-with-github-copilot>

VSCode :

Ctrl+I

Une fonction qui calcule l'intervalle de temps en minutes entre 2 dates en Javascript

New file

Ctrl+I

Calcul en temps réel d'une balance à partir d'un flux de transaction avec KafkaStream

Ouverture d'un projet SpringBoot avec VSCode, assistance à l'écriture d'une entité d'un service

TDD

Reprendre le projet ***tdd/delivery-service*** et l'ouvrir dans votre IDE

Y exécuter la commande :

```
./mvnw clean quarkus:dev
```

Accéder ensuite à l'URL <http://localhost:8000> puis suivre le lien DevUI et Continuous Testing

Exécuter les tests.

Dans votre IDE, faites échouer le test regarder le résultat sur la page de la DevUI

Écrire une autre classe de test

MergeRequest Gitlab

Mise en place

Se créer un compte sur ***gitlab.com***

Créer un projet sans l'initialiser avec un dépôt et inviter des collègues à collaborer avec le rôle *développeur*

Milestones, Issues, Labels, Tableaux de bord

En tant que mainteneur de projet, créer 2 milestones :

- **Sprint1**
- **Sprint2**

Au niveau projet, définir les labels suivants :

- **In progress**

- **Review**

Définir ensuite un tableau de bord ajoutant des colonnes pour les 2 labels précédents

Rajouter les labels

- **API**
- **DevOps**

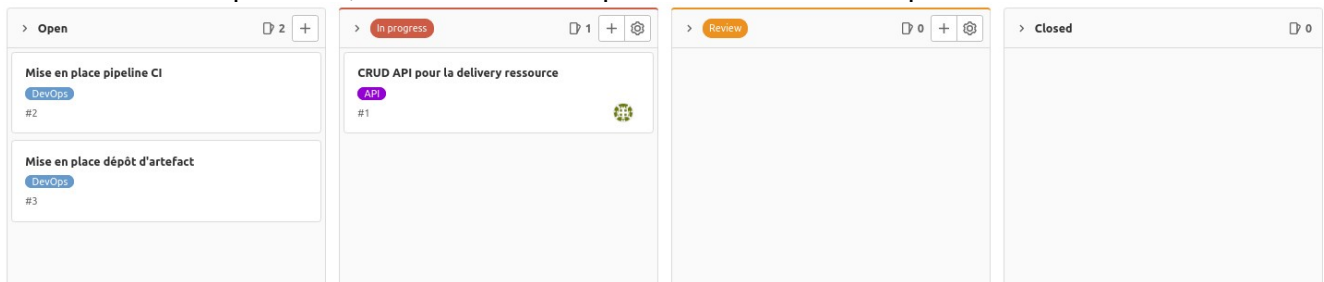
Création d'issues

- Saisir plusieurs issues dont une s'appelant : « *CRUD pour delivery-service* »
- Tagger avec *API*

Avec un compte *developer*, accès au tableau de bord et déplacement du post-it «*CRUD pour delivery-service* »

To Do -> *In progress*

A la fin de ces opérations, le tableau de bord pourra ressembler à ce qui suit :



MergeRequest et GitlabFlow

1. Création de merge request sur gitlab

En tant que développeur sur gitlab, à partir de l'issue, '*CRUD pour delivery-service*', créer une Merge Request

=> La merge request est préfixée par **Draft** et a pour effet de créer une branche portant le nom de l'issue

2. Mise en place environnement de développement + développement

En tant que développeur sur votre poste de travail :

- Si vous en avez une installer une clé ssh
- Récupérer la branche de la merge request :
`git clone <url-ssh-depot>`
`git checkout <nom-de-branche>`

Reprendre le projet **gitlab/gitlab-delivery-service**

Construire l'application :

```
./mvnw clean package
```

Exécuter l'application :

```
java -jar target/delivery-service-0.0.1-SNAPSHOT.jar \
--spring.profiles.active=swagger
```

Accéder à l'application :

<http://localhost:8080/swagger-ui.html>

<http://localhost:8080/actuator>

3. Pousser les modifications

Le développeur pousse les modifications

git add .

git commit -m 'Implémentation CRUD'

git push

En tant que *developer* sur gitlab, supprimer le préfixe *Draft*

4. Revue de code

En tant que *owner/mainteneur*, faire une revue de code en utilisant l'onglet MR

Refuser la MR

4. Développeur

Accéder à la revue de code et faites des modifications en conséquence refaire un commit et un push

5. Mainteneur

Visualiser la MR et ses différents onglets

CI/CD

Dans le répertoire **jenkins**, modifier le script de lancement selon votre environnement.

L'exécuter dans Git Bash

Se logger avec **admin/admin**

Démarrer Sonarqube

```
docker run -d --name sonarqube -p 9000:9000 sonarqube
```

Accéder à SonarQube en tant qu'admin/admin sur <http://localhost:9000>

Se créer un token (*Profile* → *Security*)

Reprendre le token dans Jenkins en mettant à jour le credential SONAR_TOKEN

Administration → *Credentials*

Créer ensuite un multibranche pipeline en configurant une source Git :

<https://github.com/dthibau/bceao-jenkins-multi-module>

Visualiser l'exécution de la pipeline avec le plugin Blue Ocean

Analyses Qualité / Vulnérabilités

Analyse statique

Sonarqube <http://localhost:9000>

Snyk :

<https://app.snyk.io/org/dthibau/project/3feb9bdc-4da9-4892-8711-7ee934d7319b>

Analyse dynamique

Visualisation rapport ZAP :

dast/2024-02-02-ZAP-Report-.html

Déploiement immuable

Démarrage d'un serveur kind

kubectl apply -f deployment/delivery-service-prod.yaml

Exposer le service :

kubectl port-forward service/delivery-service 8080:8080

Accéder à <http://localhost:8080>

Observabilité

Démo grafana plbsi

<http://217.182.253.17:3000/d/sOae4vCnk/spring-boot-statistics>