Ateliers

Formation : La pratique DevOps en environnement Java

Pré-requis :
Poste développeur avec accès réseau Internet libre
Linux (Recommandé) ou Windows 10
Pré-installation de :

- JDK8
- Git
- Docker
- Oracle VirtualBox
- Distribution minikube de Kubernetes

Manipulations pour visualiser les solutions :

- `git clone https://github.com/dthibau/delivery-service.git`
- `mv delivery-service solutions-delivery-service`
- `mkdir delivery-service`
- `cd  delivery-service`
- `git init`

Pour chaque projet ou il y a un tag, il suffit d'appeler le script ***goto.sh*** pour mettre à jour le projet delivery-servuce avec la solution de l'atelier :

```
cd solutions-delivery-service
./goto.sh <tag>
```

=> Le projet ***delivery-service*** est alors dans l'état du tag

Atelier2: SCM et Workflow de collaboration *GitlabFlow*

1. En tant que développeur sur gitlab, à partir d'une issue, créer une Merge Request
=> La merge request est préfixée par WIP et a pour effet de créer une branche portant le nom de l'issue

2. En tant que développeur sur son poste de travail, Récupérer la branche de la merge request, (*git clone + git checkout*)

Reprendre le tag 2.1 des solutions

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. Effectuer un push sur la branche
4. En tant que *developer* sur gitlab, supprimer le préfixe *WIP*
5. En tant que *owner*, faire une revue de code et ajouter comme commentaire : « Et les tests ? »

Reprise du tag 2.2 des solutions

```
Exécuter les tests et s'assurer qu'ils passent :
./mvnw test
```

6. Push les modifications vers gitlab
7. En tant que Owner faire une revue de code
8. Accepter le Merge Request et supprimer la branche
9. En local, en tant que développeur supprimer la branche locale et exécuter
git remote prune origin

Atelier 3 : Maven : Outils de Build

3.1 Graphe de tâches

Dans le répertoire de Maven *$HOME/.m2/*
Créer un fichier *settings.xml* avec les lignes suivantes :

```
<settings>
  <pluginGroups>
        <pluginGroup>fr.jcgay.maven.plugins</pluginGroup>
  </pluginGroups>
</settings>
```

Afficher le graphe de tâches avec

```
./mvnw buildplan:list
```

3.2 Gestion des versions

Quelle est la version d'hibernate utilisée ?

Quelle est la dépendance fixant toutes les versions ?

3.3 Analyse statique du code, Intégration de Sonar

- Démarrage sonar :

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

- Création d'une MergeRequest 'Mettre en place Sonar'

- Mise en place côté dév.
- `git fetch origin, git checkout`

Reprise du tag 3
Visualisez les différences via le *pom.xml* et
`./mvnw buildplan:list`
Démarrer une analyse avec
`./mvnw clean verify`

Atelier 4 – Nexus : Dépôts d'artefacts

Démarrer un serveur Nexus via Docker

```
docker volume create --name nexus-data
docker run -d -p 8081:8081 --name nexus -v nexus-data:/nexus-data
sonatype/nexus3
```

Présentation des dépôts gérés par Nexus, en particulier ***maven-central, maven-public, maven-releases, maven-snapshot***

Reprise du tag 4

Effectuer un déploiement vers le dépôt des snapshots

Utiliser le plugin Maven Release puis vers le dépôts de release. Cela nécessite de configurer le dépôt dans le *pom.xml*

Atelier 5 – Jenkins : Plateforme CI/CD

5.1 Installation Jenkins
Télécharger une distribution .war de Jenkins
Démarrer le serveur dans un terminal via
`java -jar jenkins.war --httpPort=8082`
Connecter vous à *localhost:8082* et finaliser l'installation en installant les plugins suggérés

Visualiser le lien Administration Jenkins et les différents menus proposés :

- Configuration système
- Crédentiels
- Configuration des outils
- Gestion des plugins
- Nœuds Esclave et exécuteurs

5.2 Mise en place d'une pipeline CI

- Créer une merge request sur gitlab *« Mise en place CI »*
- Création dans Jenkins d'un job ***« Multi-branch Pipeline »*** et configurer les sources du job vers le dépôt Gitlab. Configuration du scan du dépôt chaque minute

- Dans l'environnement projet :
  - **Reprendre le tag 5.2**
  - Visualiser le *JenkinsFile* et le compléter
  - Dans la branche de features, committer puis push
- Attendre que le pipeline s'exécute
- Éventuellement fixer vos erreurs

Reprendre le tag 5.3 et comparer avec votre solution

Atelier 6 – Vagrant, Ansible : Gestion de configuration

6.1 Mise à disposition des serveur d'intégration via *vagrant*
Reprendre le tag 6.1
Visualiser le fichier vagrantfile, générer une paire clé-publique/clé-privé et l'ajouter dans le dossier

Démarrage des machines virtuelles Vagrant
```
vagrant up
vagrant ssh
ssh vagrant@192.168.99.2
ssh vagrant@192.168.99.3
```

6.2 Mise en place d'un playbook Ansible
Installer ansible
Reprendre le tag 6.2
Vérifier la connexion d'ansible aux 2 machines virtuelles

Visualiser les changements dans *pom.xml* en particulier le profil prod
Visualiser le playbook ***ansible/delivery.yml*** et essayer de compléter ce qu'il manque

Exécuter le playbook
```
cd ansible
ansible-playbook delivery.yml -i hosts
```

Vérifier le bon déploiement de l'application
Accès à l'appli via :
http://192.168.99.2:8080/swagger-ui.html

Reprendre le tag 6.3 et comparer avec votre solution

6.3 Pipeline CI : Déploiement vers serveurs d'intégration

Intégrer l'appel du playbook à la pipeline Jenkins, si la branche est différente de *master*

Exécution de la pipeline et observer le bon déploiement de l'application

6.4 Tests post-déploiement

Reprendre le tag 6.4 et visualiser la pipeline Jenkins (ajout de 2 phases : Tests fonctionnels et tests de performance)

Exécuter la pipeline

Récupérer et visualiser le rapport de performance

Atelier 7 : Pipeline CD, Déployer une Release

Reprendre le tag 7

Observer les changements sur la pipeline.

Exécuter la pipeline sur la branche master

Effectuer une release

Voir également : https://plugins.jenkins.io/scmskip/

Atelier 8 : Docker

8.1 Familiarisation docker, docker-compose
Quelques commandes docker

Reprendre le tag 8.1

Visualiser le fichier *src/main/docker/postgres-docker-compose.yml*

Démarrer la stack et créer une base de donnée via *pgAdmin*

8.2 Mis en place d'un *docker-compose* pour l'application en profile prod

Reprendre le tag 8.2
Visualiser les changements dans :

- *src/main/docker/docker-compose.yml*
- *src/main/resources/application.yml*
- *pom.xml*

Test le profil de *prod* dans l'environnement de Développement

Atelier 9 : Pipeline CD avec image Docker

9.1 Mise en place des dépôts Docker

Voir : https://blog.sonatype.com/using-nexus-3-as-your-repository-part-3-docker-images

Déploiement latest et release

9.2 Intégration dans pipeline Jenkins

Intégrer le déploiement d'images Docker vers le registre nexus dans la pipeline.

Supprimer le déploiement via Ansible et effectuer les tests JMeter en démarrant une stack via *docker-compose*

Nécessite l'installation du plugin Performance

Reprendre le tag 9.2

Comparer votre solution

Atelier 10 - Kubernetes

Démarrage *minikube* ou *microk8s*

Accès au dépôt privé Nexus : voir https://kubernetes.io/fr/docs/tasks/configure-pod-container/pull-image-private-registry/

kubectl create secret generic regcred
--from-file=.dockerconfigjson=/home/dthibau/.docker/config.json
--type=kubernetes.io/dockerconfigjson

10.1 : Déploiements à partir d'une image

Créer un déploiement à partir d'une image docker

```
kubectl create deployment delivery-service
--image=nexus:18082/delivery-service:0.0.1-SNAPSHOT
```

Exposer le déploiement via un service

```
kubectl expose deployment delivery-service --type LoadBalancer \
  --port 80 --target-port 8080
# Vérifier exécution des pods
kubectl get pods
# Accès aux logs
kubectl logs <pod_id>

kubectl get service delivery-service
#Forwarding de port
kubectl port-forward service/delivery-service 8080:80
```

Accès à l'application via localhost:8080

Mise à jour du déploiement

```
kubectl set image deployment/delivery-service delivery-
service=nexus:18082/delivery-service:0.0.3-SNAPSHOT
```

Statut du roll-out
```
kubectl rollout status deployment/delivery-service
```

Accès à l'application : *http:<IP>/actuator/info*

#Visualiser les déploiements
```
kubectl rollout history deployment/nginx-deployment
```

#Effectuer un roll-back
```
kubectl rollout undo deployment/delivery-service
```

```
#Scaling
kubectl scale deployment/delivery-service --replicas=5
```

10.2 : Déploiement d'une stack SB/Postgres

Reprendre le tag 10.2

Visualiser les fichiers du répertoire *src/main/k8*
Déployer la stack et accéder à l'application, l'utiliser

Atelier 11 – Pipeline avec Kubernetes

11.1 Un environnement par Merge Request

But avoir un environnement de recette par feature branch/MergeRequest

11.2. Roll-out de la production

Dans la branche master, mettre à jour un déploiement

Reprendre le tag 11.1 et comparer