Java EE Spring, prêt à l'emploi CI/CD

Guillaume Dufrêne – Lionel Seinturier – Julien Wittouck

Université de Lille

Plan

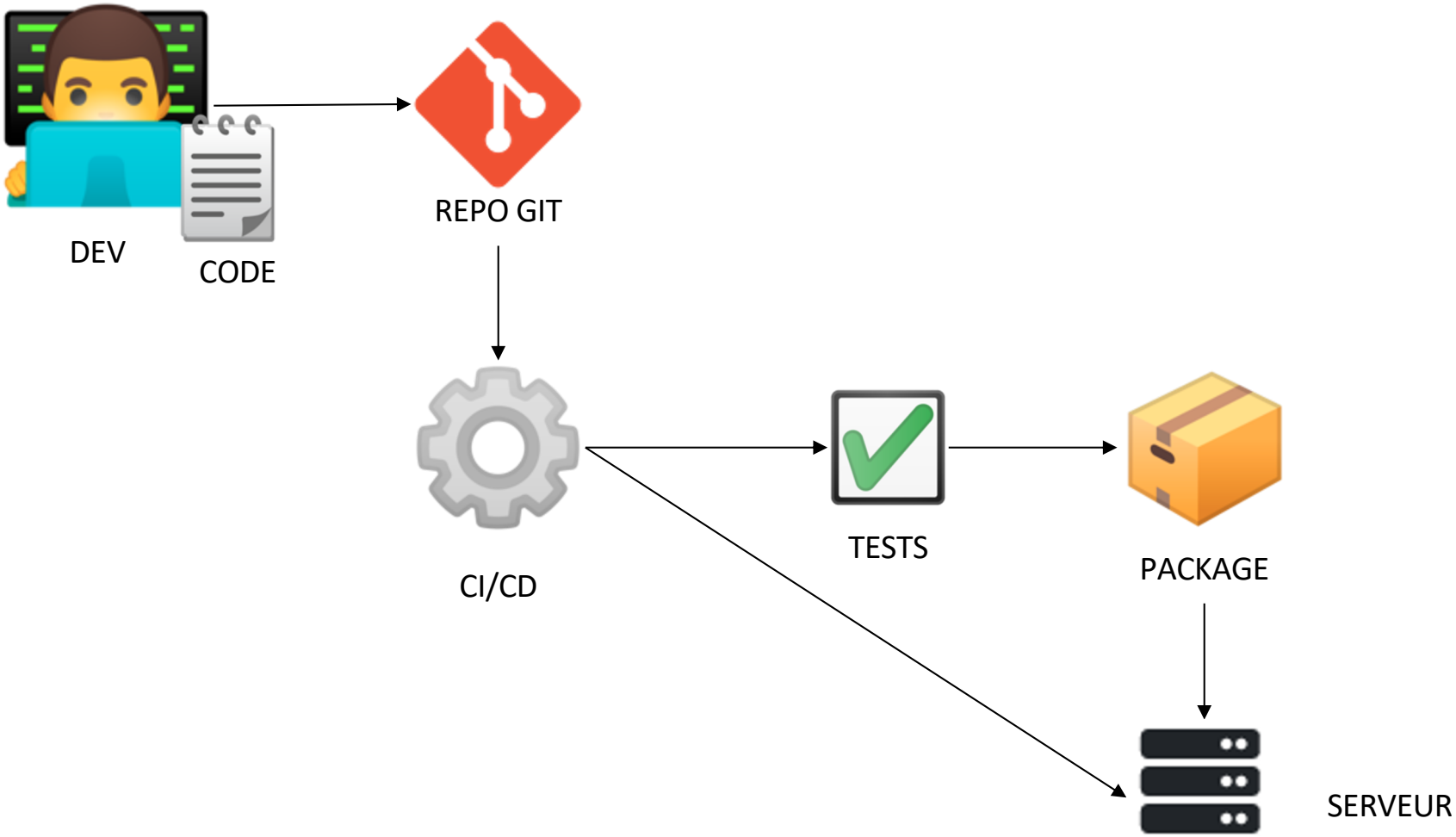
- Principes de l'intégration continue et du déploiement continu
- Notion de pipeline
- Outils d'intégration continue
- Exemple de pipeline avec Gitlab-CI

Introduction

- L'intégration continue et le déploiement continu (CI/CD)

- l'intégration continue (CI):
  - intégrer rapidement le code des développeurs
  - valider la non-régression du code
  - exécuter les tests unitaires
  - suivre l'évolution de la qualité du code
- le déploiement continu (CD):
  - déployer de manière automatique les applications
  - mettre à disposition rapidement des nouvelles versions de l'applications
  - monter d'environnement en environnement

La vision globale



Notion de pipeline

- Le pipeline constitue l'ensemble des étapes d'une intégration continue

CI
- compilation
- exécutions des tests unitaires / d'intégration
- analyse de qualité de code et de couverture de tests
- analyse de sécurité (dépendances)
- production d'un binaire (jar/war/image docker)

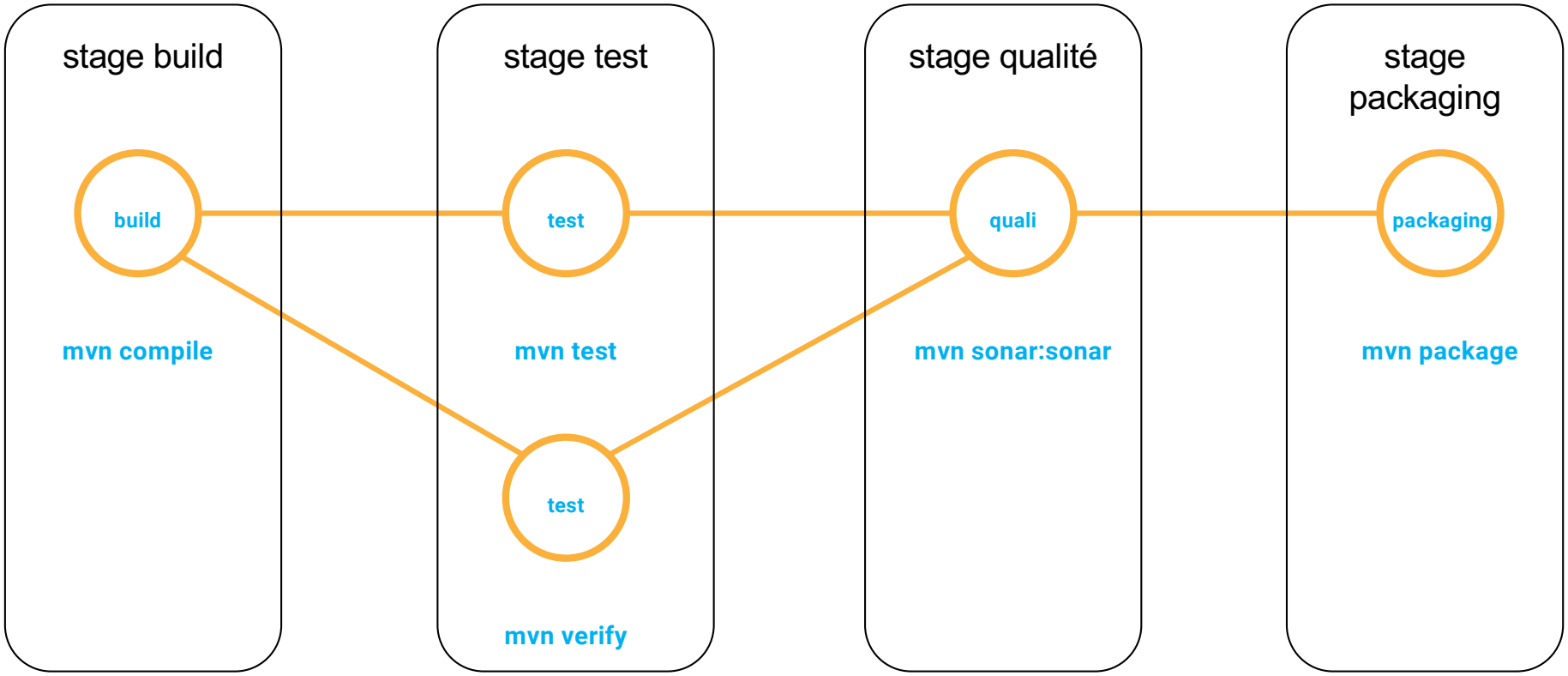
CD
- déploiement sur un environnement de développement
- déploiement sur un environnement de staging
- déploiement sur un environnement de production

Pipelines

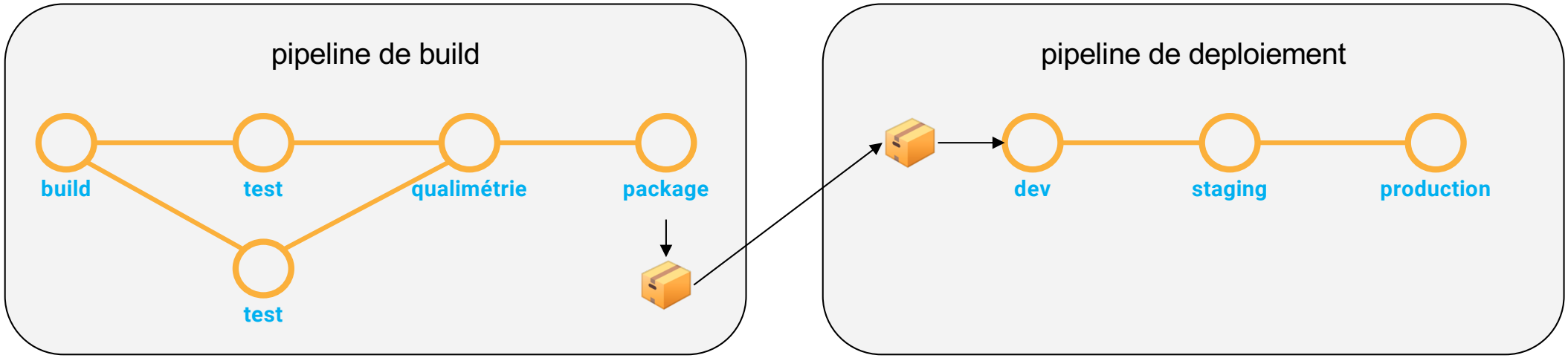
Notion de pipeline (concepts communs à toutes les plateformes)

Découpé en stage. Les jobs d'un stage sont exécutés en parallèle.

Un pipeline est exécuté pour chaque commit/branche



Enchaînement de pipelines et passages d'artefacts



Le pipeline de build est déclenché lorsque du nouveau code est disponible.
Le pipeline de déploiement est déclenché lorsqu'un nouveau package est disponible.

Au plus près du code

- Gitlab-CI : Moteur d'intégration continu intégré à Gitlab
  - Pipelines définis en yaml
  - Pipeline "Auto DevOps" pour déploiement sur des environnements Kubernetes
  - Déclenchement sur des modifications de code ou schedulés
- Github Actions : Moteur d'intégration continu intégré à Github
  - Pipelines définis en yaml
  - Déclenchement sur des modifications de code et de projet (création d'issues, application d'un label ou d'une milestone)

Externalisés

- Jenkins
  - Pipelines définis en Groovy
  - Système de plugins
  - Self-Hosted
- travis-ci
  - Pipelines définis en yaml
  - Intégration avec Github/Gitlab
  - Orientation vers l'open-source et les projets publics (gratuité)
- ...

Exemple avec Gitlab-CI

```yaml
stages:
  - build
  - test
  - package

build:
  stage: build
  image: maven:3-jdk-11
  script:
    - mvn compile test-compile

test-jdk-11:
  stage: test
  image: maven:3-jdk-11
  script:
    - mvn test

test-jdk-15:
  stage: test
  image: maven:3-jdk-15
  script:
    - mvn test

package:
  stage: package
  image: maven:3-jdk-11
  script:
    - mvn package
  only:
    - main
  artifacts:
    paths:
      - target/*.war
```

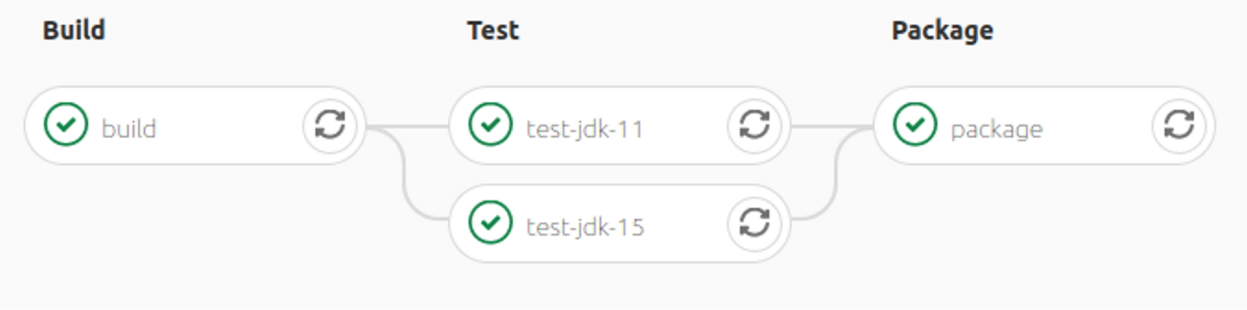
Les stages du pipeline

Chaque job s'exécute dans une image docker

Le job build compile le code et les tests

Les jobs "test" exécutent les tests dans 2 images docker différentes : jdk 11 et 15

Le job package produit le war et l'enregistre comme artefact



Conclusion

L'intégration continue et le déploiement continu permettent de :

- construire régulièrement l'application
- obtenir un feedback rapide sur les tests et la qualité du code
- automatiser le packaging et le déploiement de l'application