Introduction aux pratiques DevOps

Adrien CAUBEL

Pourquoi ce cours ?

- Nous avons vu les pratiques Agiles
- Nous avons un cours dédié aux pratiques d'intégration

- Les pratiques Agile et DevOps sont étroitement liées
  - DevOps applique les principes Agiles en dehors du développement
  - Sont des pratiques complémentaires

Naissance du mouvement

Histoire

- 2001 – Manifeste Agile
  - Adaptation aux changements
  - Livraison fréquente / Itérations
  - Satisfaction client
  - Auto-organisation des équipes
  - Apprentissage continue / Feedback
- 2009 – Le mouvement DevOps

Pourquoi ce mouvement ?

- On a des silos de compétences
  - Equipe de développement / Equipe d'exploitation

- Le mouvement vise donc
  - Eliminer ces silos
  - Améliorer la collaboration et la communication entre ces deux groupes
  - Renforcer l'efficacité du processus de livraison des logiciels

Agile, Lean, DevOps : complémentaires

- DevOps et Agile visent toutes deux à améliorer de développement et la distribution de logiciel

- DevOps applique les principes Agile et Lean à l'ensemble de la chaîne logistique logicielle.
  - Permet à une entreprise d'optimiser la rapidité de livraison d'un produit ou d'un service
  - D'assurer un feedback

- On cherche a réduire le gaspillage
  - Réduire les goulots d'étranglement
  - Utilisation d'outils pratiques pour automatiser des processus qui étaient traditionnellement manuels et lents (e.g. déploiement du code)

Agile, Lean, DevOps : complémentaires

« On retrouve le vocabulaire évoqué dans le cours sur Agile »

Agile, Lean, DevOps : différences

- Agile (définition)
  - Pratique incrémentale pour développer un logiciel
  - Dans le but de fournir de la valeur au client
  - En s'adaptant rapidement au changement

- DevOps (définition)
  - Intégrer les équipes d'exploitation dans le développement de logiciel
  - Cette équipe déploie et distribue le logiciel aux utilisateurs finaux
    - Suivre les changements du logiciel
    - Gérer les modifications de configuration

DevOps

Development

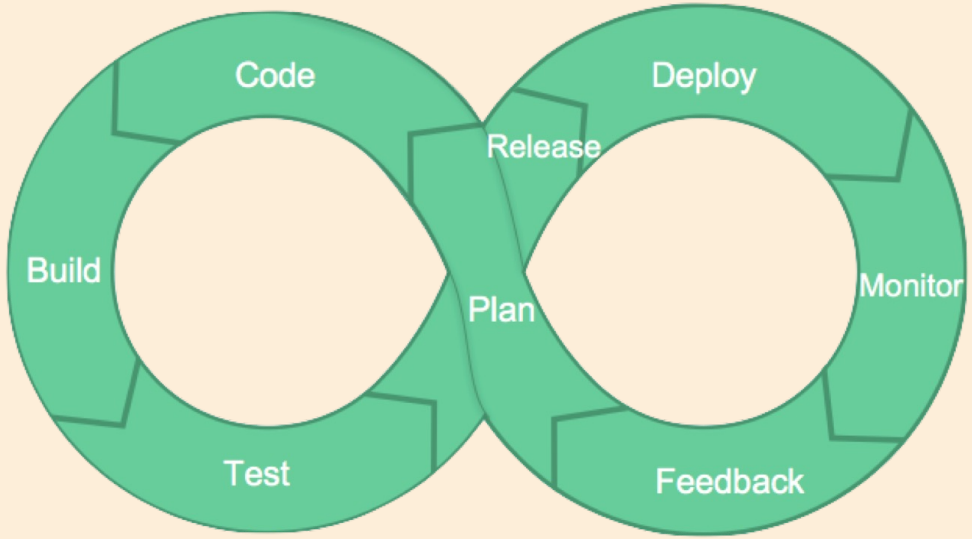
Operations (fr = exploitation)

DevOps élimine les silos et met l'accent sur la collaboration entre l'équipe de développement et l'équipe d'exploitation.

DevOps

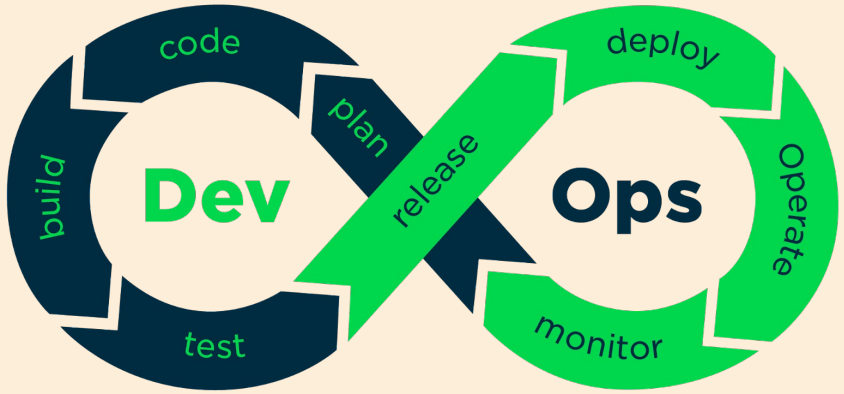
Définition

- C’est une industrialisation de l’informatique
- Avec une amélioration continue via une boucle de feedback



Feedback loop devops que l’on voit partout sur internet

Feedback loop



- Objectifs
  1. Améliorer le produit
  2. **Tout en** améliorant le process qui permette de livrer le produit régulièrement

- Il faut donc parcourir cette boucle le plus rapidement
  - Valeur au client == déploiement
  - Pour être sur de ne rien louper, il y a différentes étapes

DevOps ce n’est pas

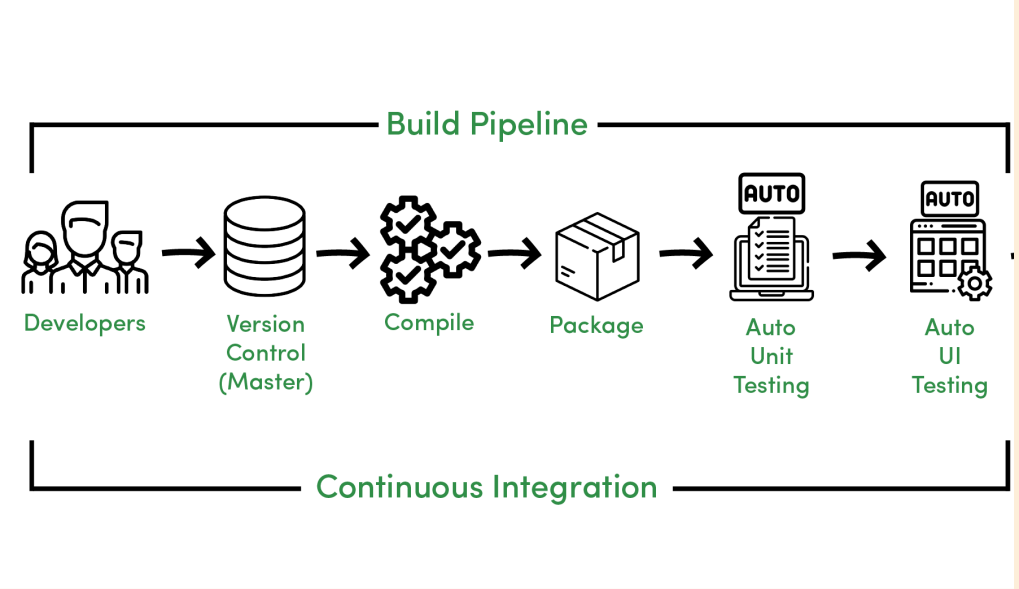
- Donc DevOps :
  - Ce n’est pas que des outils
    - Docker
    - Kubernetes
    - Jenkins
    - Ansible
    - …
- Le DevOps c’est un principe pour améliorer notre logiciel de manière continue

Principes DevOps

https://blog.stephane-robert.info/post/devops-une-refondation-n%C3%A9cessaire/

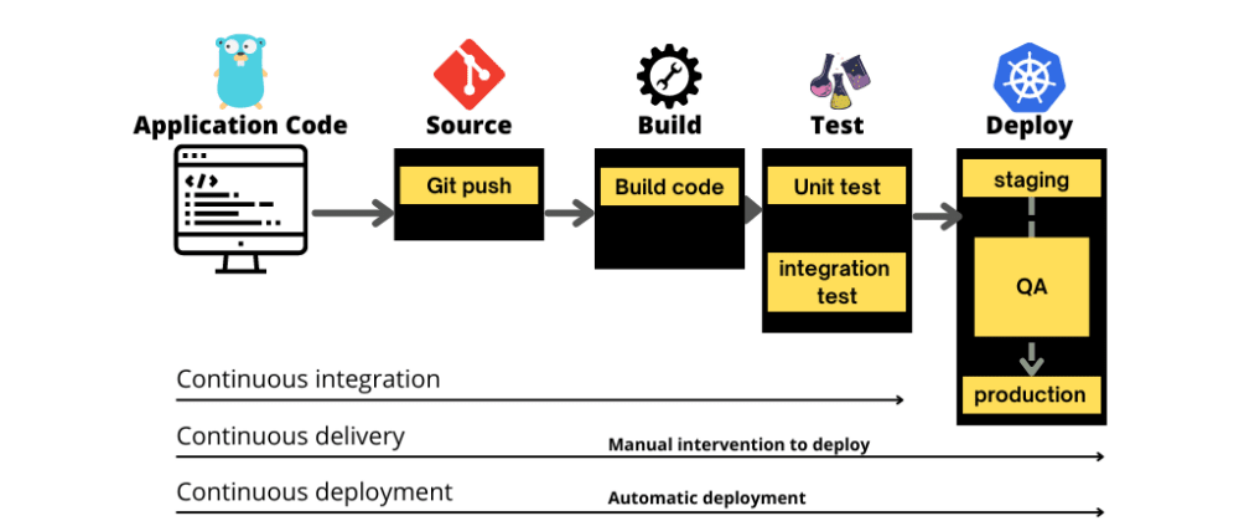
CI/CD

Continious Intégration (CI)



- Pouvoir intégrer facilement et rapidement du code de différents développeurs

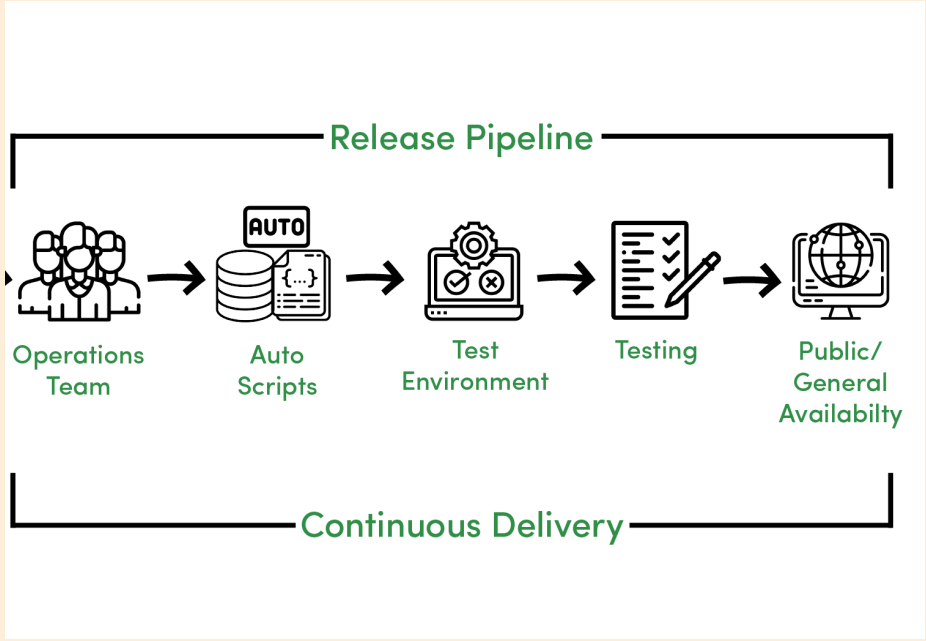
Continious Intégration (CI)



1. Vous avez votre code en local (sur votre pc)
2. Vous le *push* sur un repository distant
3. Ceci va lancer un build : vérifier que votre code compile avec le code de master
4. Ceci va lancer des tests (e.g. Junit)

5. *Votre code est mis dans un environnement de déploiement*

Continious Delivery (CD)



- Pouvoir déployer l'application chez le client rapidement et « automatiquement »

Continious Delivery (CD)

1. Vous avez votre code en local (sur votre pc)
2. Vous le *push* sur un repository distant
3. Ceci va lancer un build : vérifier que votre code compile avec le code de master
4. Ceci va lancer des tests (e.g. Junit)

Environnement de Développement

5. *Votre code est mis dans un environnement de production*
6. Le code est testé sur un environnement de production

Environnement de Production

7. *Votre code est prêt à être déployé chez le client*

---

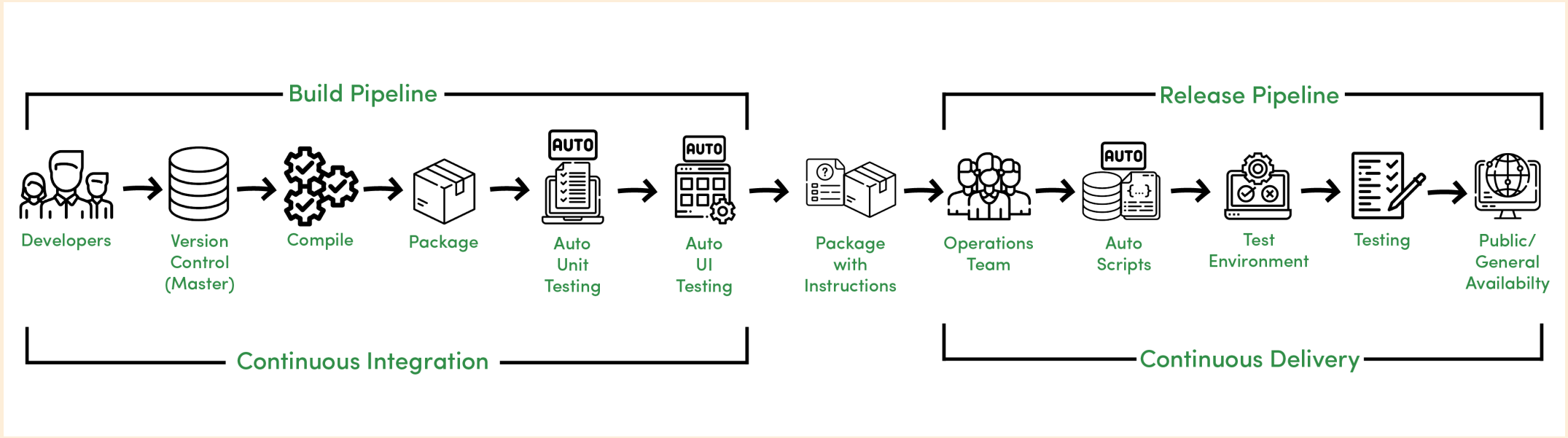
8. On récupère du feedback

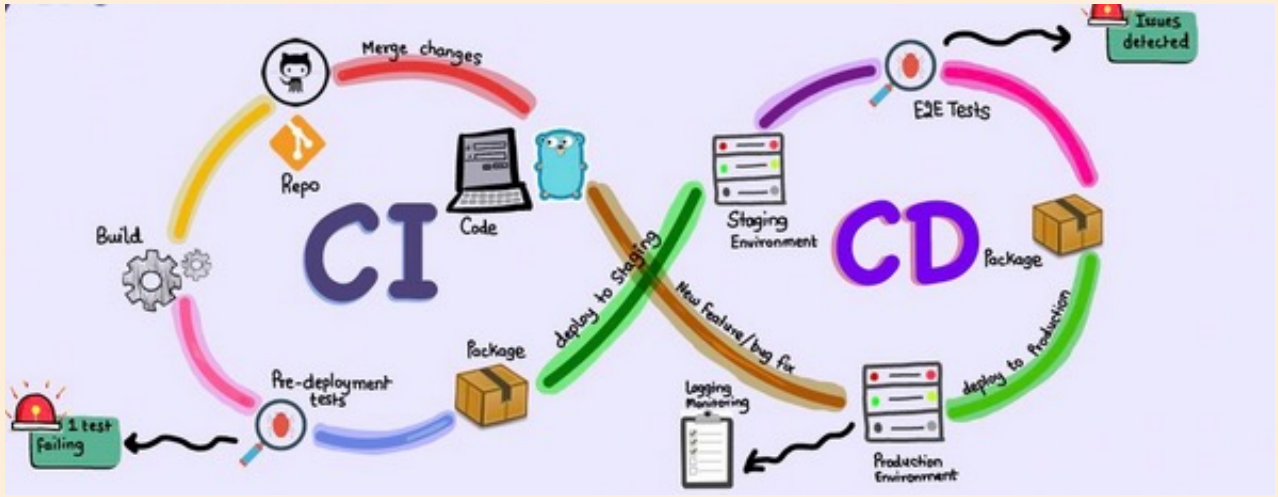
Chez le client

Pipeline CI/CD

- CI = lancer le build + test
- CD = lancer le déploiement

Pipeline CI/CD





Pipeline CI/CD : outils CI

- Travis CI, Jenkins CI, Buildkite, GitHub Actions

- Se lancent sur chaque commit
- Et on obtient une « manière » juste = qui peut être déployée

Pipeline CI/CD : outils CD

- Harness, Vercel, Heroku, GitHub Pages/Actions

- Permet de déployer des versions de code automatiquement au fur et à mesure que l'on travaille dessus.
  - Déploie généralement une branche git **release** vers la production, et d'autres branches git vers un environnement de développement

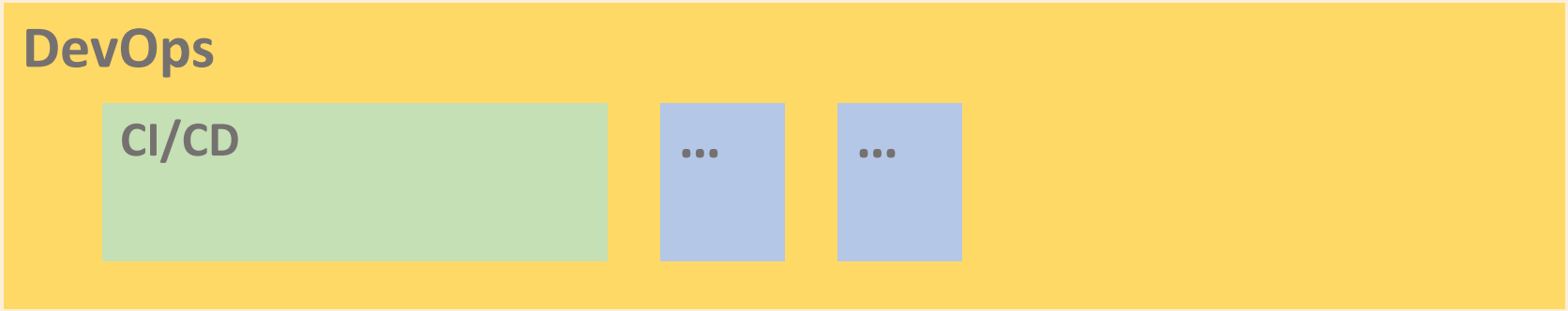
DevOps et CI/CD

DevOps et CI/CD ??

- DevOps est une culture et un processus visant à rendre le développement de logiciels plus efficace.

- Un pipeline CI/CD est un ensemble spécifique de phases liées à des outils et à des automatisations qui permettent de réaliser le cycle de vie DevOps

- Un pipeline CI/CD est intrinsèquement lié à une culture DevOps



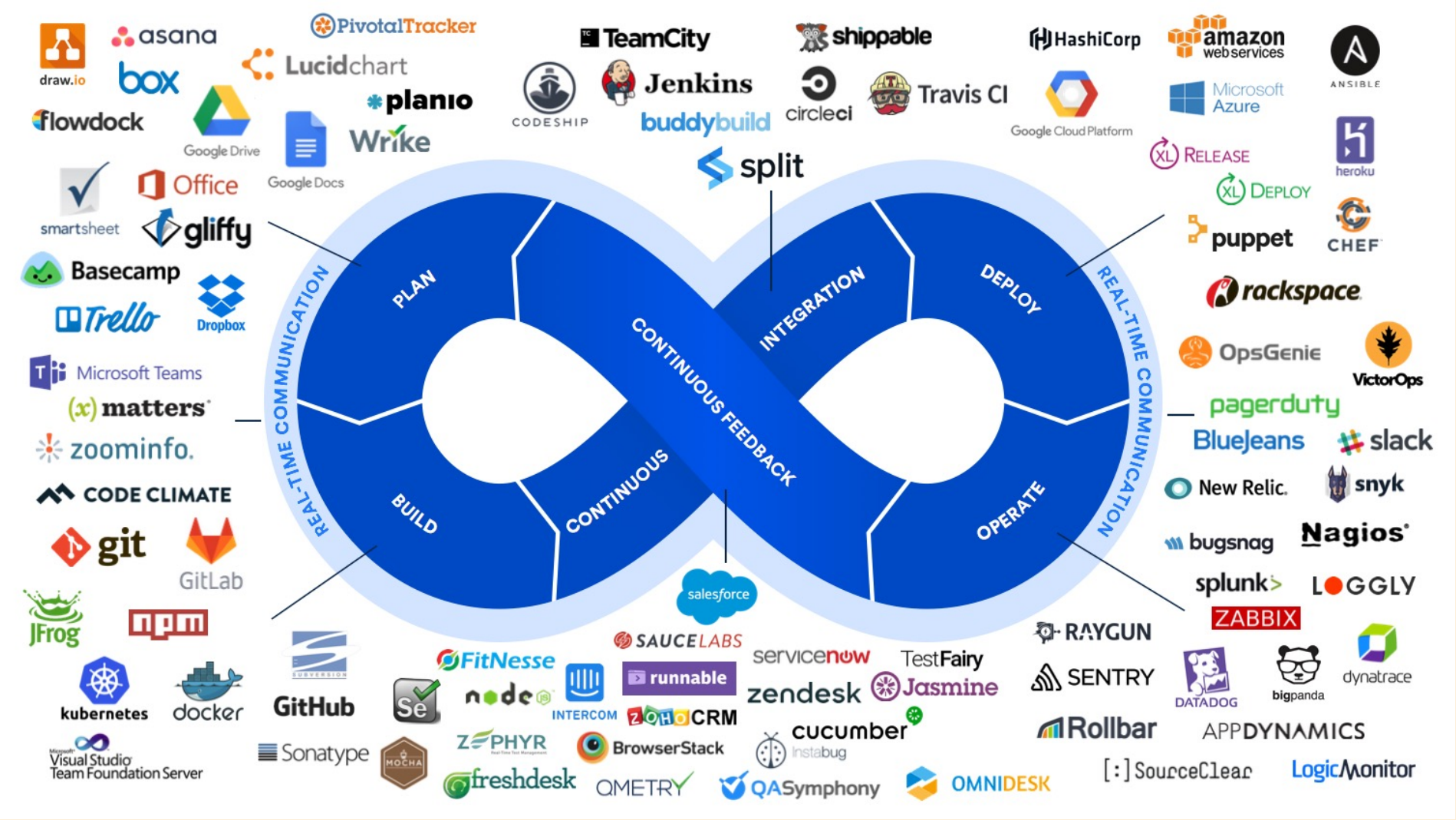
Outils

Les outils

- DevOps != Outils (comme Agile != framework Scrum, Kanban, …)

- Les outils vont nous aider à mettre en œuvre les pratiques DevOps
  - Nous allons voir quelque outils qui aide la mise en place de la pipeline CI/CD

Enormément d'outils …

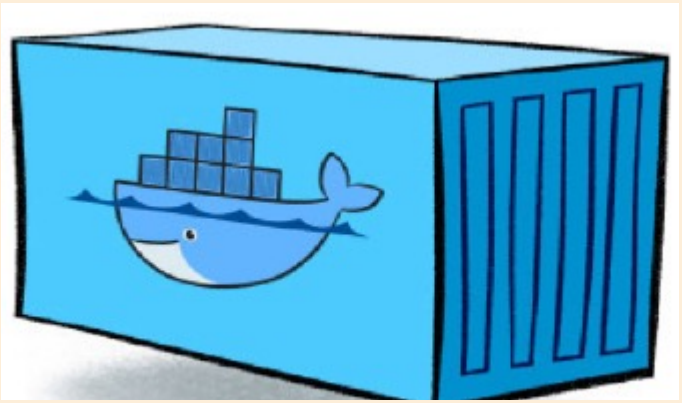


Docker

Docker

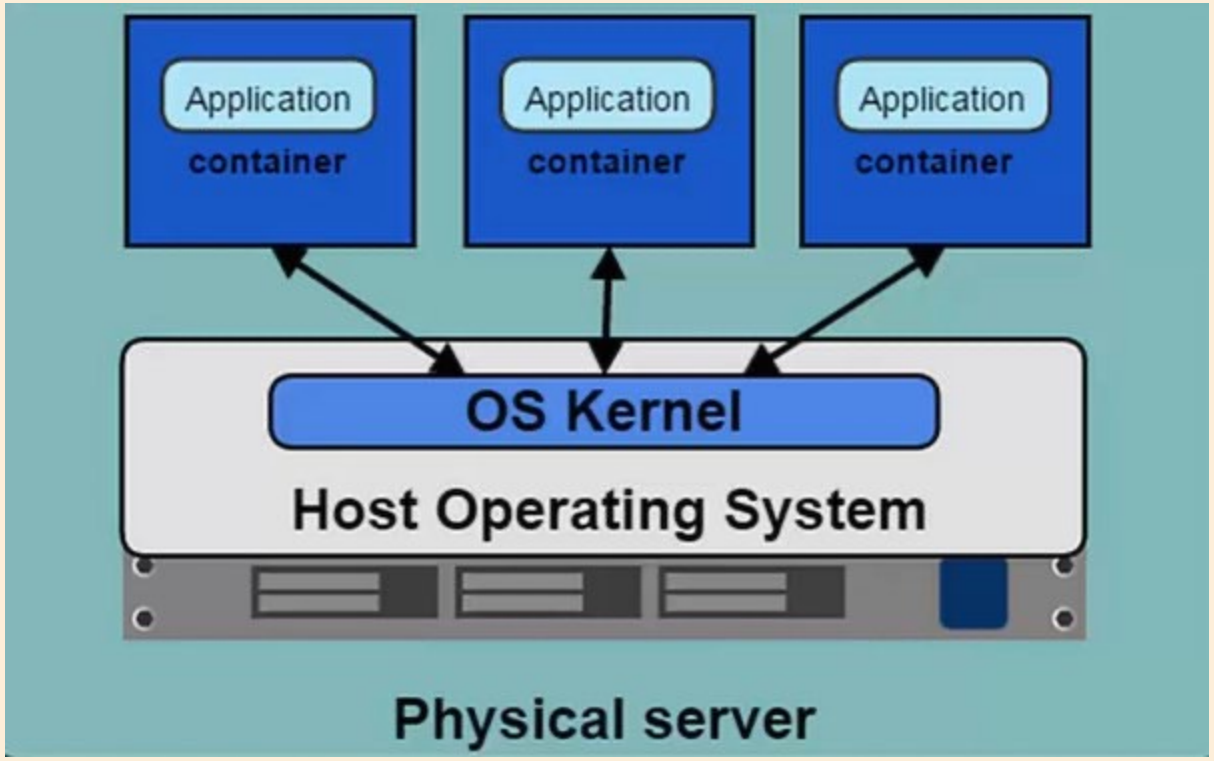
- Permet une distribution facile
  - Plutôt que de virtualiser l'ensemble du système d'exploitation, il continue d'utiliser le noyau/système d'exploitation de l'hôte comme "base" pour servir ce qui s'exécute à l'intérieur du système
- Plus rapide qu’une VM
- Conçu pour être éphémère
  - Les conteneurs sont "jetables" - toutes les données à long terme doivent être stockées dans des "volumes"

Docker



- Permet une distribution facile
  - Plutôt que de virtualiser l'ensemble du système d'exploitation, il continue d'utiliser le noyau/système d'exploitation de l'hôte comme "base" pour servir ce qui s'exécute à l'intérieur du système
- Plus rapide qu’une VM
- Conçu pour être éphémère
  - Les conteneurs sont "jetables" - toutes les données à long terme doivent être stockées dans des "volumes"

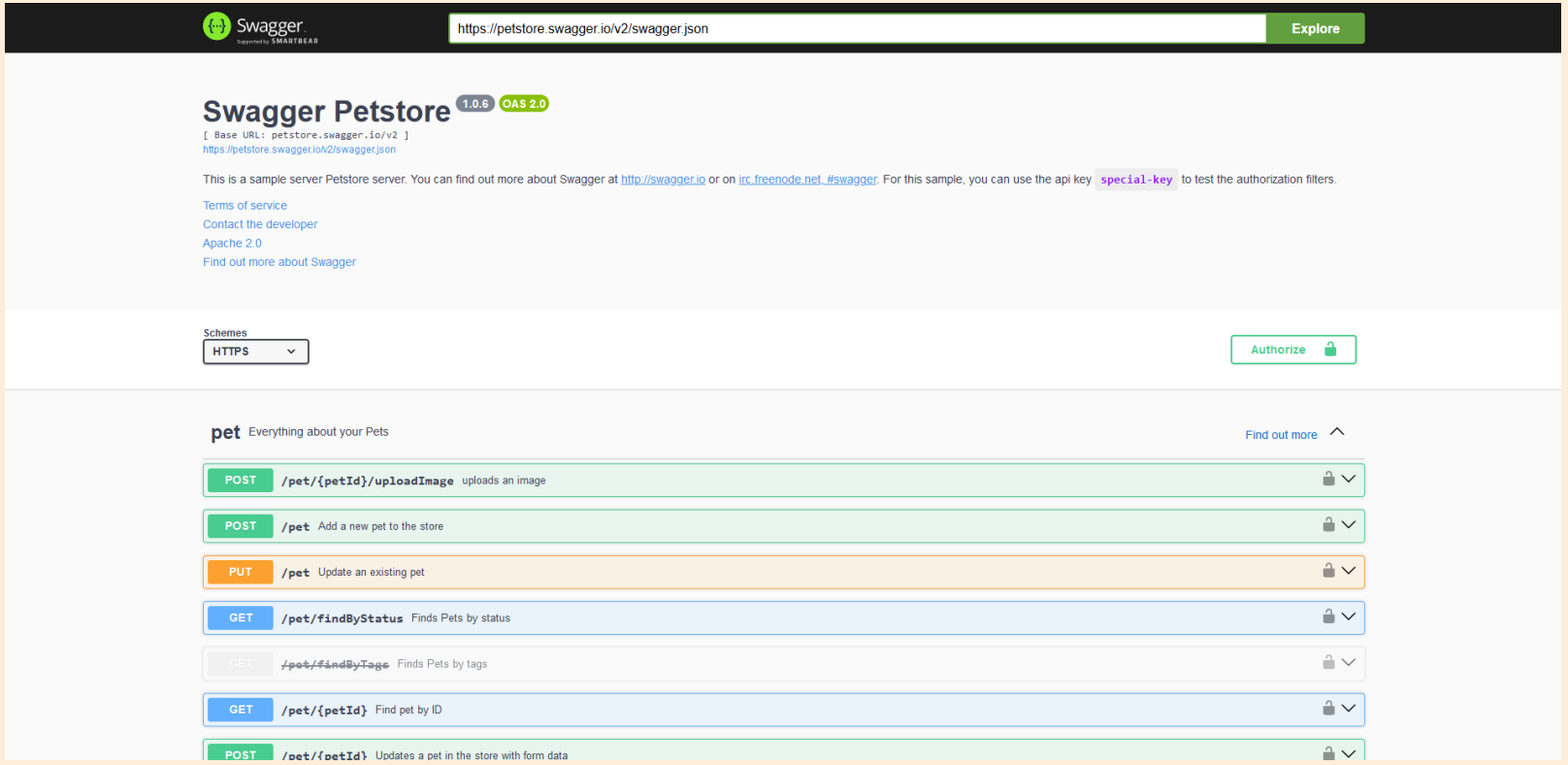
Docker


Application
container
Application
container
Application
container
OS Kernel
Host Operating System
Physical server

Exemple

```
docker pull swaggerapi/swagger-ui
docker run -p 80:8080 swaggerapi/swagger-ui
```

Sur l'ordinateur je vais sur http://localhost:80



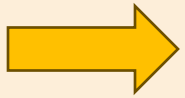
https://www.baeldung.com/java-dockerize-app

Dockerfile

Pour les SAE au lieu de rendre un .jar vous pouvez rendez un docker

```java
public class HelloWorld {
    public static void main(String[] args){
        System.out.println("Welcome to our application");
    }
}
```

```xml
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-jar-plugin</artifactId>
    <version>${maven-jar-plugin.version}</version>
    <configuration>
        <archive>
           <manifest>
              <mainClass>fr.adriencaubel.HelloWord</mainClass>
              </manifest>
         </archive>
     </configuration>
</plugin>
```

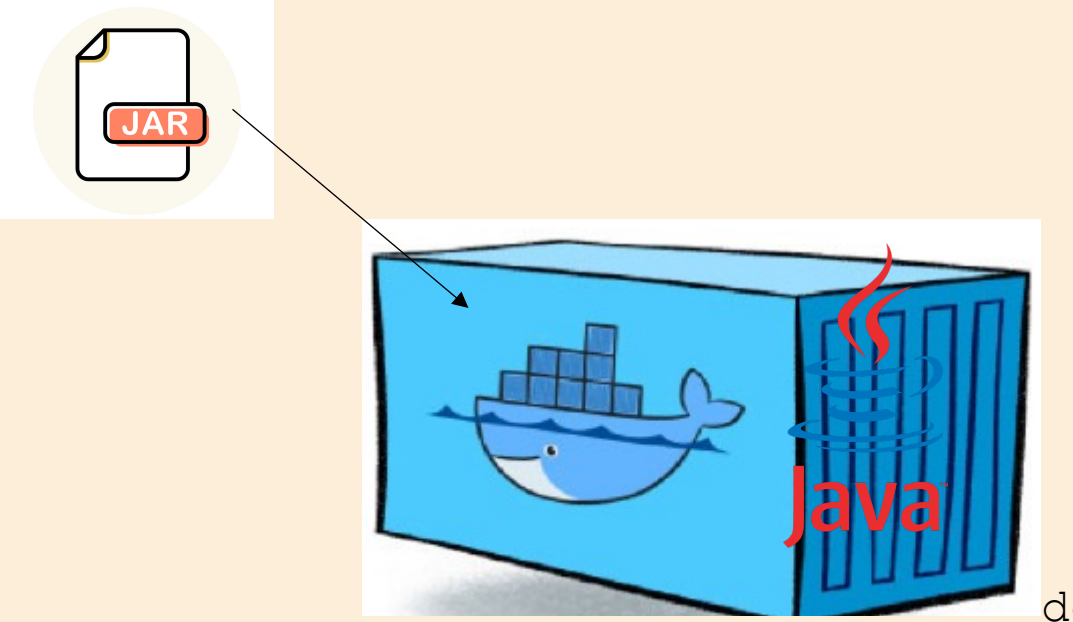


`target/docker-java-jar-0.0.1-SNAPSHOT.jar`

Le plugin maven-jar-plugin va nous créer un jar exécutable dans le dossier target/

Dockerfile

```
FROM openjdk:11
COPY target/docker-java-jar-0.0.1-SNAPSHOT.jar app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```



- nous créons une nouvelle couche en copiant le jar généré, `docker-java-jar-0.0.1-SNAPSHOT.jar`, du dossier target/ dans le dossier racine de notre conteneur avec le nom app.jar.
- nous spécifions l'application principale avec la commande de lancement

```
docker image build -t docker-java-jar:latest .
docker run docker-java-jar:latest
```

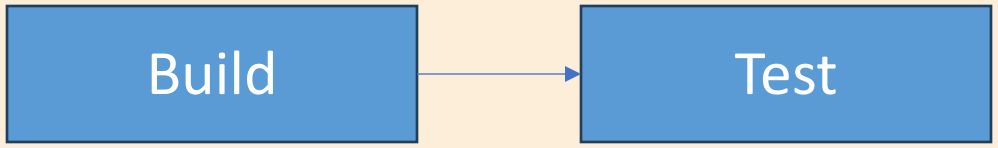
Jenkyll

Jenkins est un outil open source de serveur d'automatisation. Il aide à automatiser les parties du développement logiciel liées au build, aux tests et au déploiement, et facilite l'intégration continue et la livraison continue

Jenkins

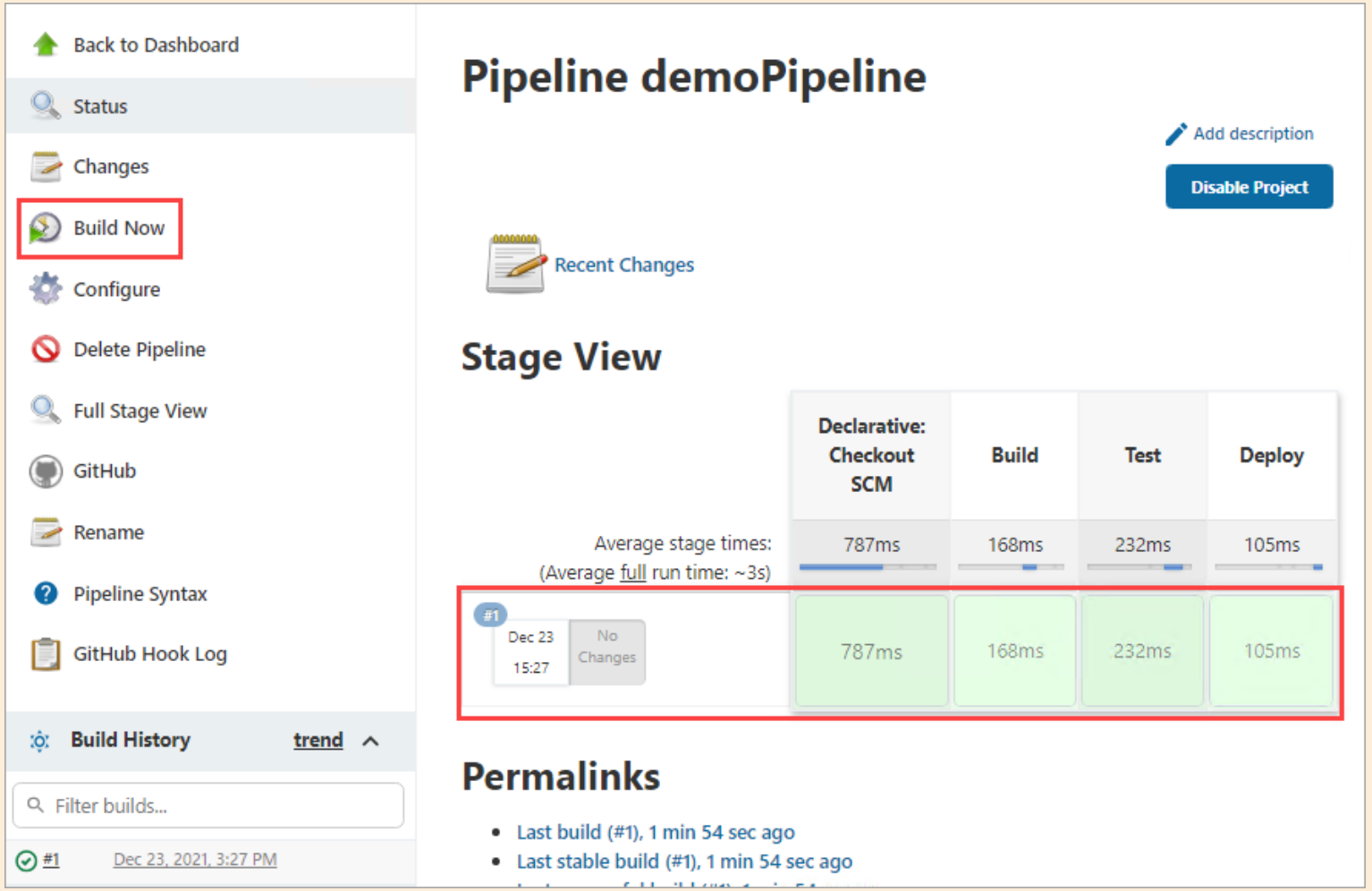
Jenkinsfile + pipeline

```
pipeline {
    agent any
    stages {
        stage('build') {
            steps {
                echo 'First Stage'
            }
        }
        stage('test') {
            steps {
                echo 'Second stage'
            }
        }
    }
}
```

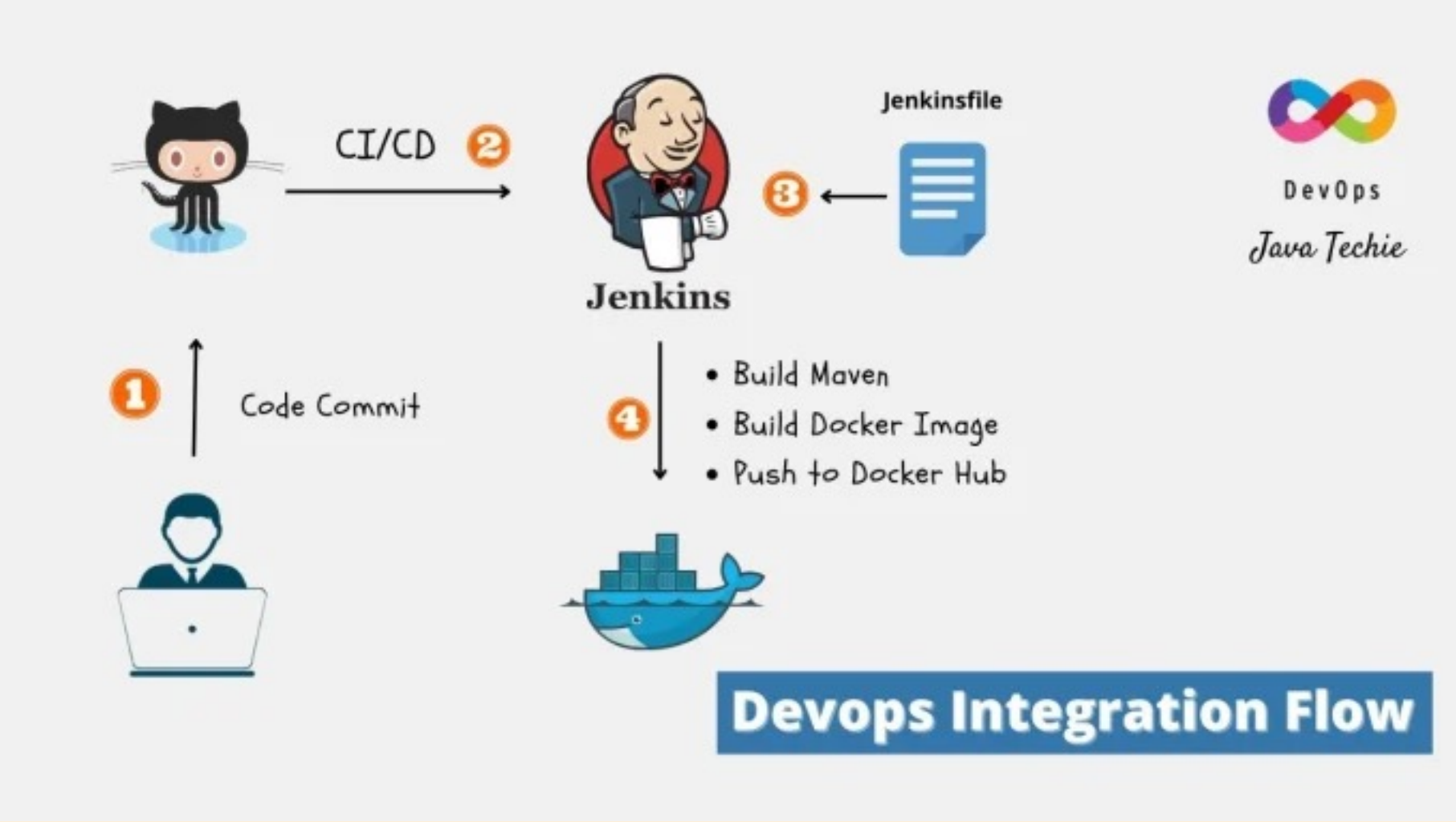


On peut également préciser un répository git où lorsqu'on push du code jenkins va automatiquement le récupérer pour lancer la pipeline

Jenkinsfile



Jenkins + Git + Docker



Spécifier à Jenkins dans le filtre « deploy » de créer une image docker