15/03/2023
Version 1

DEVOPS

JÉRÉMY PERROUAULT

CI

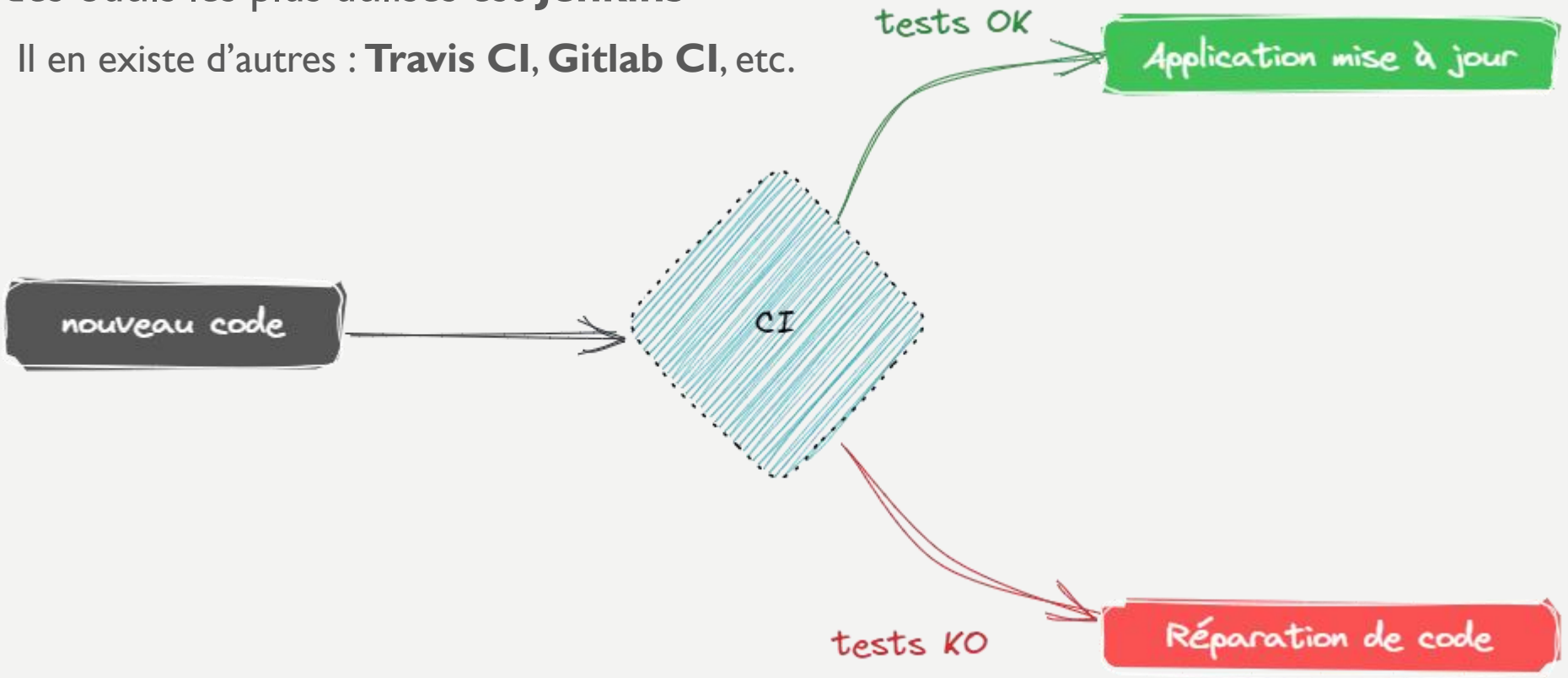
INTÉGRATION CONTINUE

INTÉGRATION CONTINUE

- Ensemble de pratiques pour tester chaque révision de code
  - De manière automatisée
  - Avant un éventuellement déploiement
- Son rôle est de garantir l'intégration d'un code de la meilleure qualité possible
  - Avec le moins d'anomalies possibles
  - Sans régressions
  - Pour une meilleure expérience utilisateur
- C'est un premier pas vers le déploiement continue (**CD**)

INTÉGRATION CONTINUE

- Un des outils les plus utilisés est **Jenkins**
  - Il en existe d'autres : **Travis CI**, **Gitlab CI**, etc.

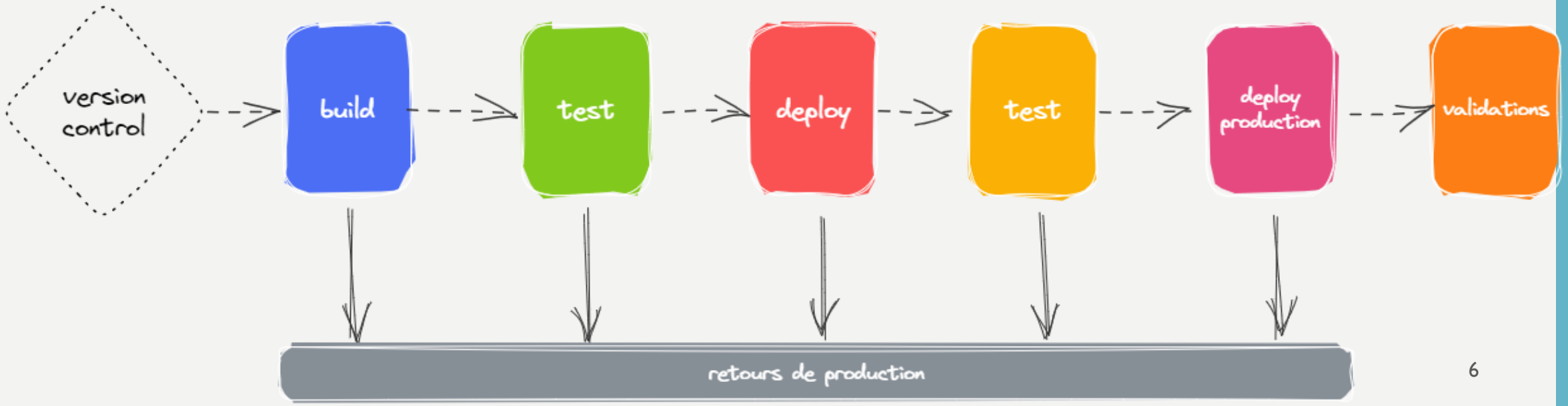


INTÉGRATION CONTINUE

- Dans **Jenkins**, on ajoute un « item »
  - Qu'on peut configurer de différentes façons
    - Pour **Maven**
    - Via un **Pipeline**
    - Etc.

INTÉGRATION CONTINUE

- Un **pipeline** est un ensemble de **stage**, dans lesquels on retrouve une ou plusieurs **steps**
  - Généralement lorsqu'une étape échoue, le processus échoue et ne va pas plus loin
- Le **pipeline** peut être directement dans le projet, dans un fichier *Jenkinsfile*
- Ou rédigé depuis l'application **Jenkins**, dans la configuration du projet



INTÉGRATION CONTINUE

- Exemple de **pipeline** pour puller un projet **git**

```
pipeline {
  agent any

  stages {
    stage('Github checkout') {
      steps {
        script {
          git credentialsId: '<credential-github-id>',
              url: 'git@github.com:<user>/<repo>.git',
              branch: 'main'
        }
      }
    }
  }
}
```

EXERCICE

- Mettre en place un *container* **Jenkins**
  - Dans le réseau « devops » avec comme IP 172.20.0.25, publier le port 8090 vers 8080
  - Suivre les étapes pour récupérer le mot de passe « admin » par défaut
  - Installer les plugins par défaut
  - Configurer **Jenkins** pour accéder à votre compte **Github** via une clé SSH

```
ssh-keygen
ssh-keyscan github.com >> ~/.ssh/known_hosts
```

- Copier la clé publique sur votre compte **Github**

```
cat ~/.ssh/id_rsa.pub
```

- Copier la clé privée sur **Jenkins** (Administrer Jenkins -> Manage credentials), avec votre `Passphrase`

```
cat ~/.ssh/id_rsa
```

EXERCICE

- Créer un projet sur **Jenkins**
- Mettre le projet **JAVA** dans **Git** (si pas déjà fait)
- Récupérer le projet **JAVA** dans **Jenkins**
  - Le build **Jenkins** doit passer !

- Se placer dans le répertoire /var/jenkins_home/workspace
  - Vérifier avec la commande « ls » si les fichiers du projet sont bien ici

INTÉGRATION CONTINUE

- Exemple de **pipeline** (extrait) pour builder et tester un projet **Maven**

```
sh 'mvn clean install'
```

- Si besoin d'exécuter **Maven** dans un sous-répertoire

```
dir('<repertoire>') {
  sh 'mvn clean install'
}
```

- Préciser également l'outil **Maven** dans le **pipeline**, dans le `stage` avant la première `step`

```
tools {
  maven 'Maven 3.9.0'
}
```

INTÉGRATION CONTINUE

- Exemple de **pipeline** (extrait) pour builder et tester un projet **Maven**
  - Via le plugin « Docker Pipeline » et l'*image* **Docker** « maven:3.9.0 »

```
stage('Maven Install') {
  agent {
    docker {
      image 'maven:3.9.0'
    }
  }

  steps {
    sh 'mvn clean install'
  }
}
```

EXERCICE

- Avec l'option « -u » Docker, exécuter *bash* en utilisateur « root »
  - Exécuter les commandes

```
apt update
apt install -y wget curl
```

  - Revenir en user jenkins
  - Se placer dans le répertoire utilisateur (/var/jenkins_home)
  - Récupérer le *tar.gz* de maven puis le dézipper avec la commande « tar »

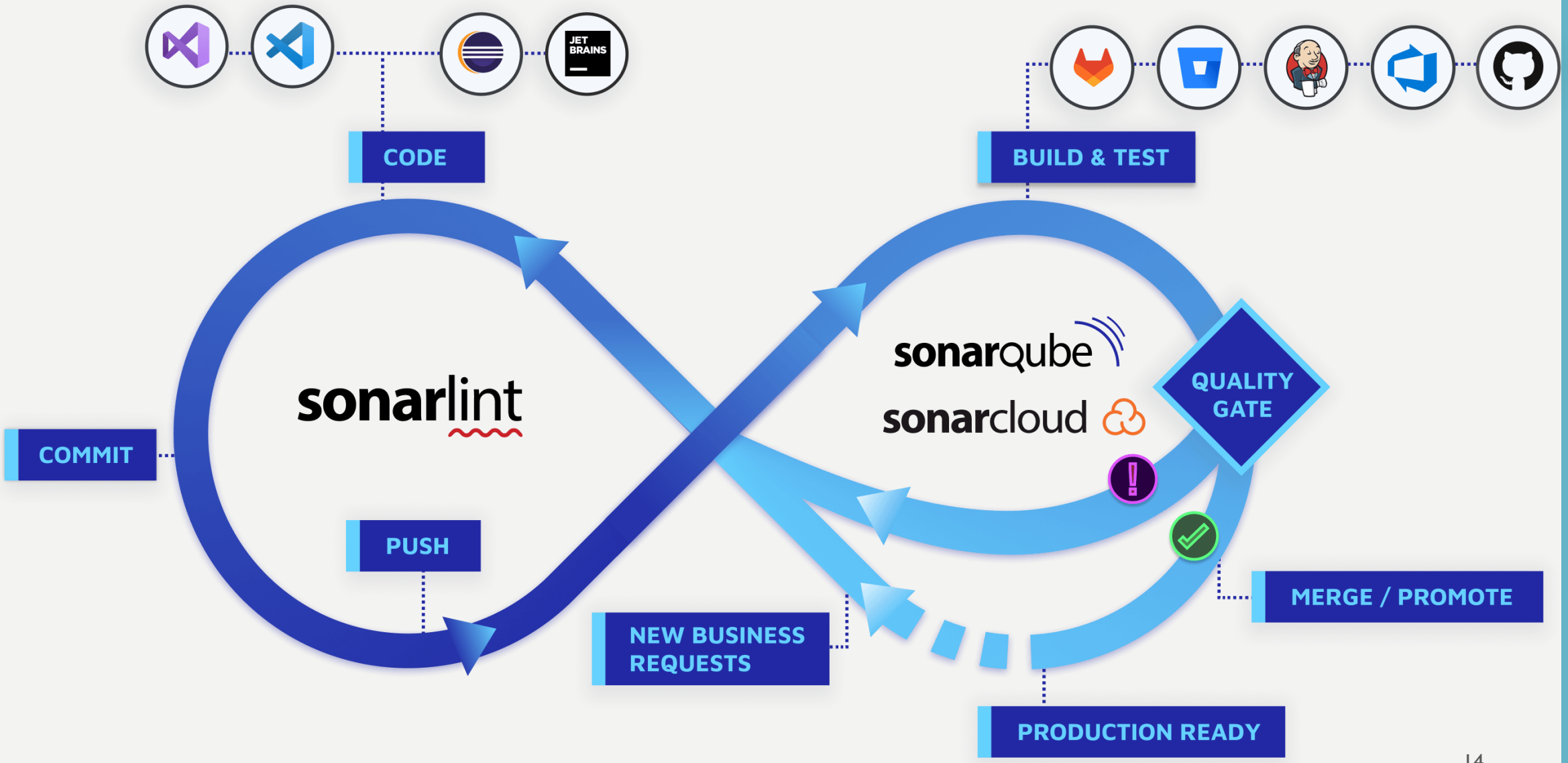
```
wget https://...apache-maven-3.9.0-bin.tar.gz
tar -xvzf apache-maven-3.9.0-bin.tar.gz
```

EXERCICE

- Dans **Jenkins**, installer le plugin « Maven Integration » puis redémarrer **Jenkins**
- Configurer le plugin **Maven** dans les outils (Administration **Jenkins**)
  - Préciser le répertoire *MAVEN_HOME* « /var/jenkins_home/apache-maven-3.9.0 »

- Ajouter une étape pour builder et tester le projet **Maven**
  - Le build **Jenkins** doit passer !

INTÉGRATION CONTINUE



INTÉGRATION CONTINUE

- Pour valider le projet avec **SonarQube**

```
withSonarQubeEnv('<IdAccesSonarQube>') {
  sh 'mvn clean verify sonar:sonar -Dsonar.projectKey=<nom_projet>'
}

waitForQualityGate abortPipeline: true
```

EXERCICE

- Dans **SonarQube**
  - Ajouter un jeton à Sonar (My Account > Generate Tokens)
  - Ajouter un Webhook
    - http://adresse-ip-jenkins:8080/sonarqube-webhook

- Dans **Jenkins**, installer le plugin « SonarQube Scanner » puis redémarrer **Jenkins**
- Dans les propriétés Système de **Jenkins**
  - Ajouter une liaison **SonarQube**
    - Avec une clé « Secret text » qui prendra pour id « SONARQUBE_TOKEN »
    - Attention ici, utilisez l'adresse IP du container **SonarQube**

- Ajouter une étape pour analyser avec **SonarQube**
  - Le build **Jenkins** doit passer !

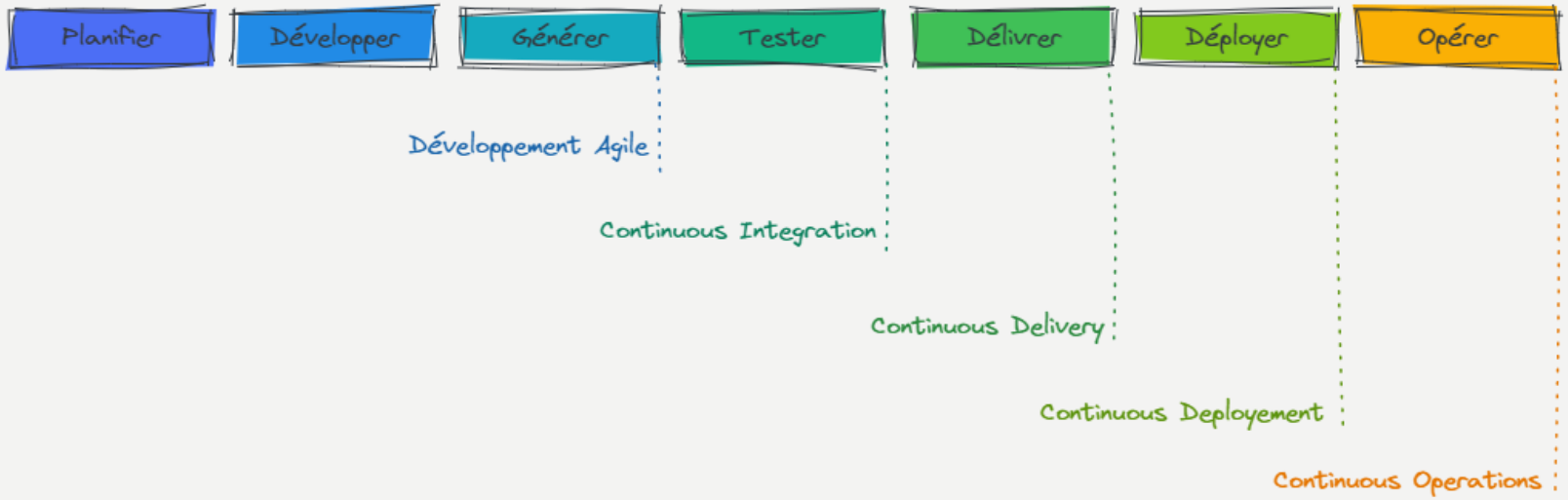
CD

DÉPLOIEMENT CONTINUE

DÉPLOIEMENT CONTINUE

- Continuité du **CI**
- Consiste à automatiser les tâches de déploiement sur un environnement
  - Pré-prod, production, etc.

- Contrairement à la livraison continue, le déploiement continue va jusqu'au déploiement



DÉPLOIEMENT CONTINUE

- Exemple de **pipeline** (extrait) pour builder sur **Docker**

```
stage('Docker Build') {
  agent any
  steps {
    sh 'docker build -t <nom_image>:<tag> .'
  }
}
```

DÉPLOIEMENT CONTINUE

- Exemple de **pipeline** (extrait) pour stopper et lancer un *container* sur Docker

```
stage('Docker Run') {
  agent any
  steps {
    script {
      try {
        sh 'docker stop <nom_container>'
      }

      finally {
        sh 'docker run --rm -d --name <nom_container> -p 5000:8080 <nom_image>'
      }
    }
  }
}
```

EXERCICE

- Avec l'option « -u » Docker, exécuter *bash* en utilisateur « root »
  - Exécuter les commandes (la dernière commande sera réinitialisée à chaque reboot de **Jenkins**)

```
apt install -y docker.io
chmod 777 /var/run/docker.sock
```

  - Revenir en user jenkins et vérifier que la commande « docker » fonctionne dans le *container*

```
docker version
docker ps
```

EXERCICE

- Dans **Jenkins**, installer les plugins « Docker » et « Docker Pipeline » puis redémarrer **Jenkins**

- Ajouter une étape pour builder l'*image* **Docker** et la conteneuriser
  - Le build **Jenkins** doit passer !