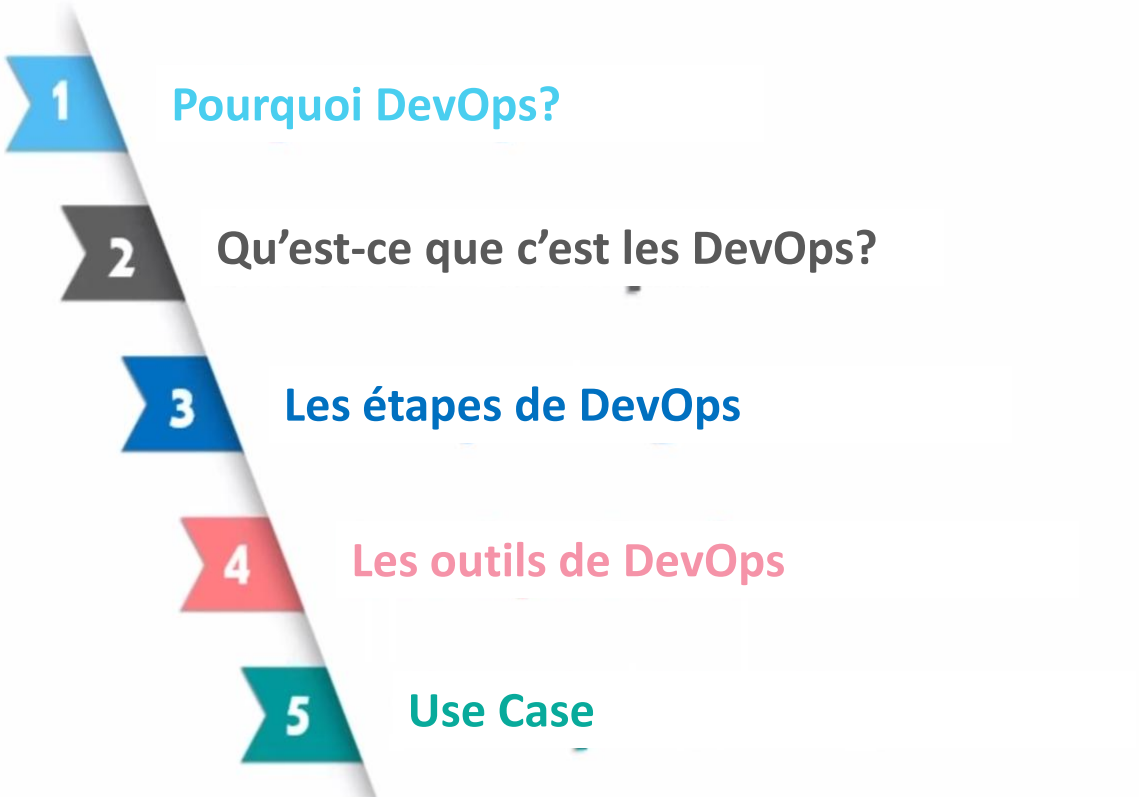
ESPRIM
Ecole Supérieure Privée d'Ingénieurs de Monastir
المدرسة العليا الخاصة للمهندسين بالمنستير

CHAPITRE 5 : INTRODUCTION AUX DEVOPS

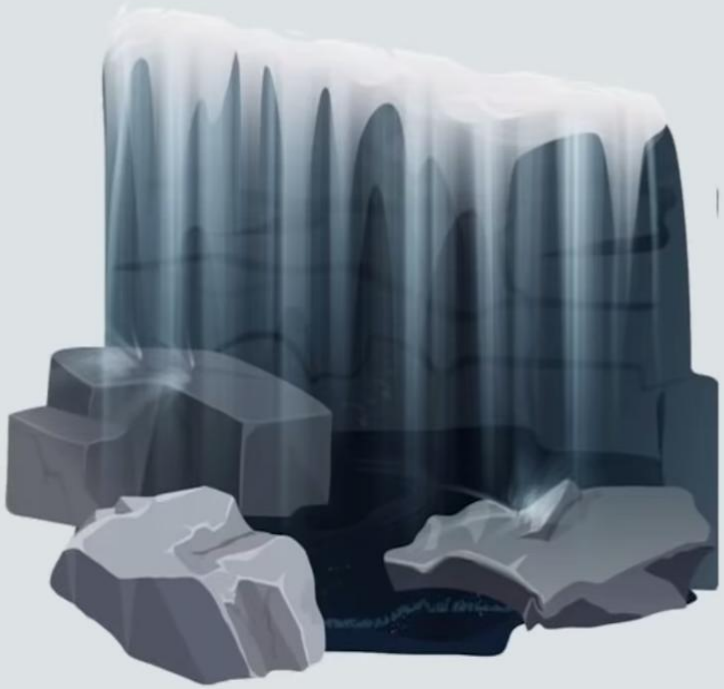
MARIEM ZAOUALI

00 SOMMAIRE

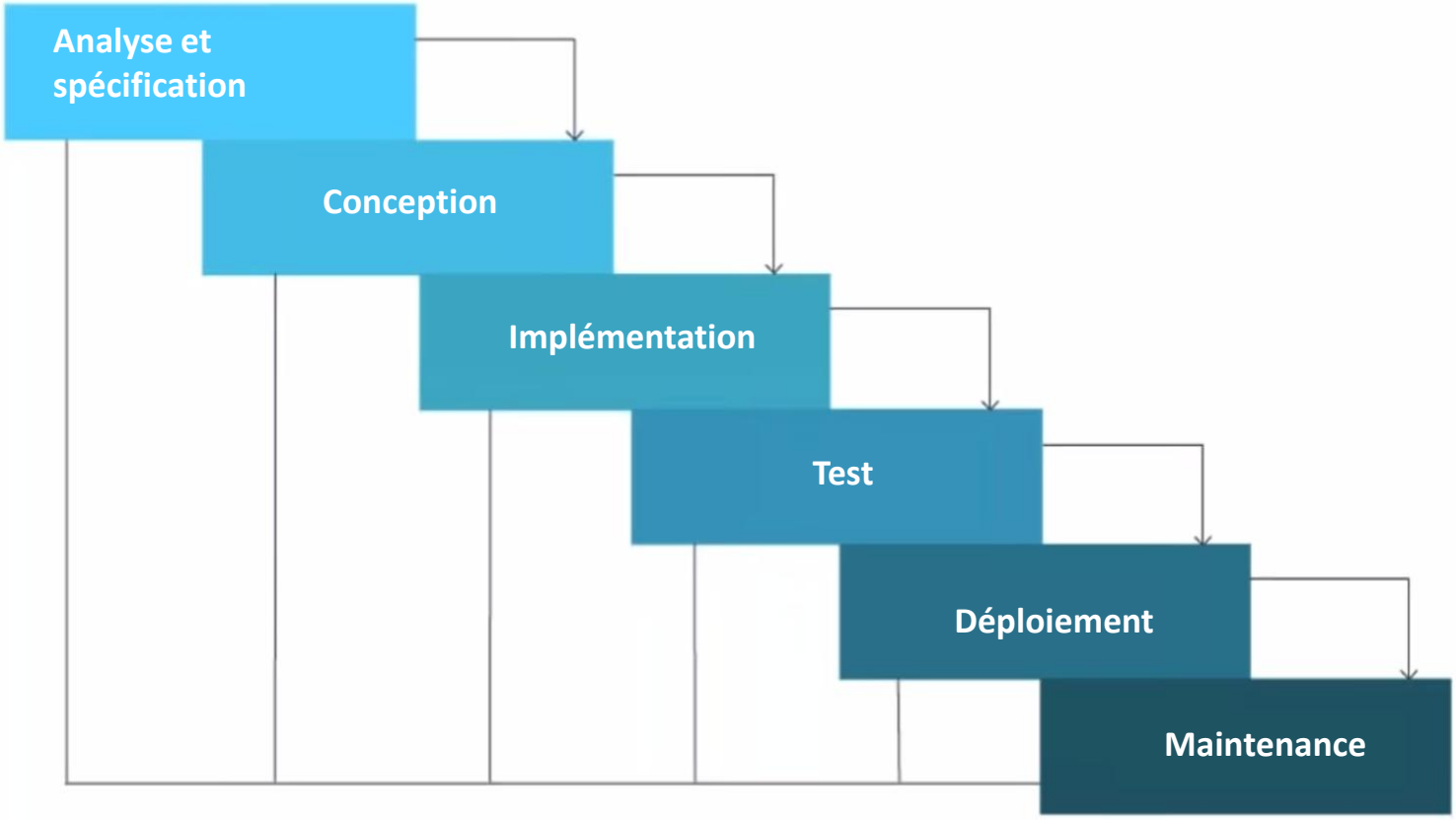

1
Pourquoi DevOps?
2
Qu’est-ce que c’est les DevOps?
3
Les étapes de DevOps
4
Les outils de DevOps
5
Use Case

Modèle en cascade (Waterfall model)

Il a différents objectifs pour chaque phase de développement. Il s’appelle ainsi car si on quitte une étape, on n’est plus en mesure de faire une marche arrière.

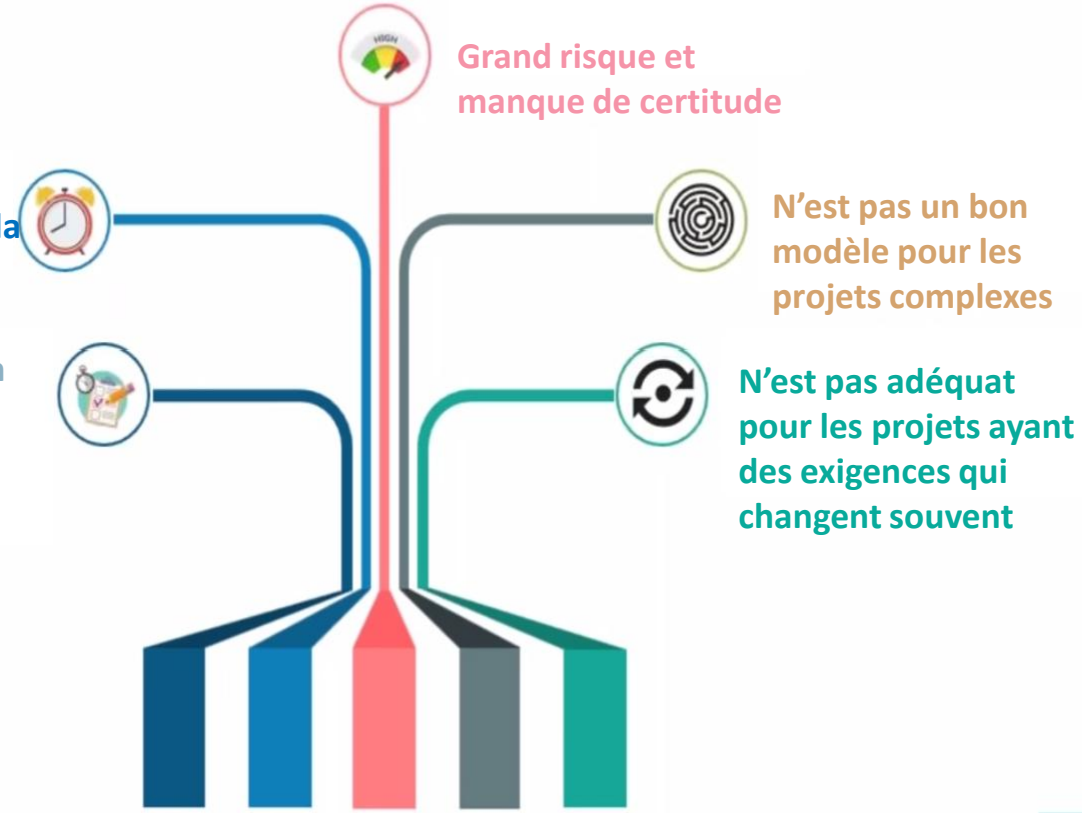


01 MODÈLE EN CASCADE TRADITIONNEL



01

LIMITE DU MODÈLE EN CASCADE



Grand risque et manque de certitude

Aucun livrable fonctionnel avant la fin du cycle

N’est pas un bon modèle pour les projets complexes

Une fois l’application est en phase de test, il est trop tard de faire marche-arrière et faire des modifications qui n’ont pas été bien méditées lors de la conception

N’est pas adéquat pour les projets ayant des exigences qui changent souvent

01 RAPPEL SUR LA MÉTHODOLOGIE AGILE

Dans la méthodologie chaque projet est divisé en plusieurs itérations

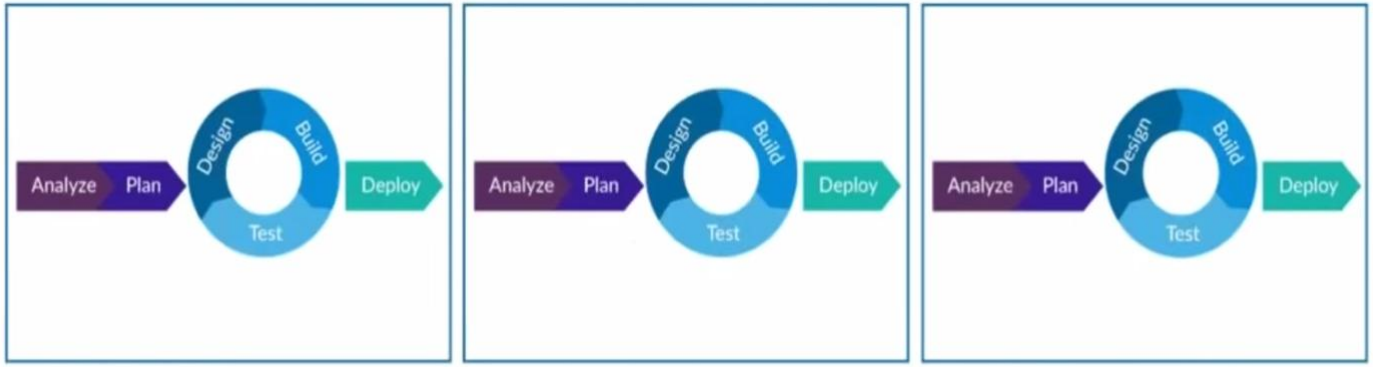
Toutes les itérations doivent avoir la même durée (entre 2 à 8 semaines)

A la fin de chaque itération, un produit de travail doit être livré

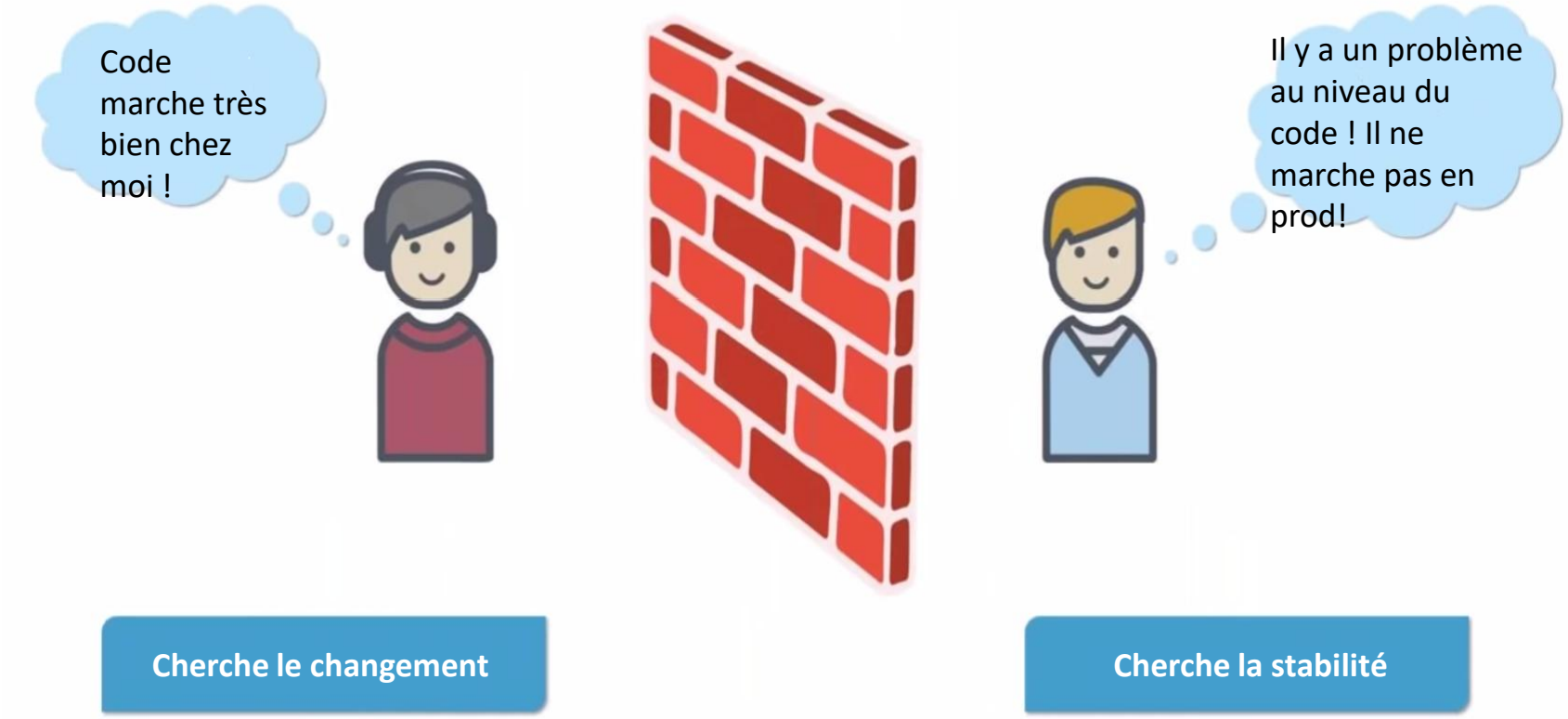
01 LIMITE DU MODÈLE EN CASCADE

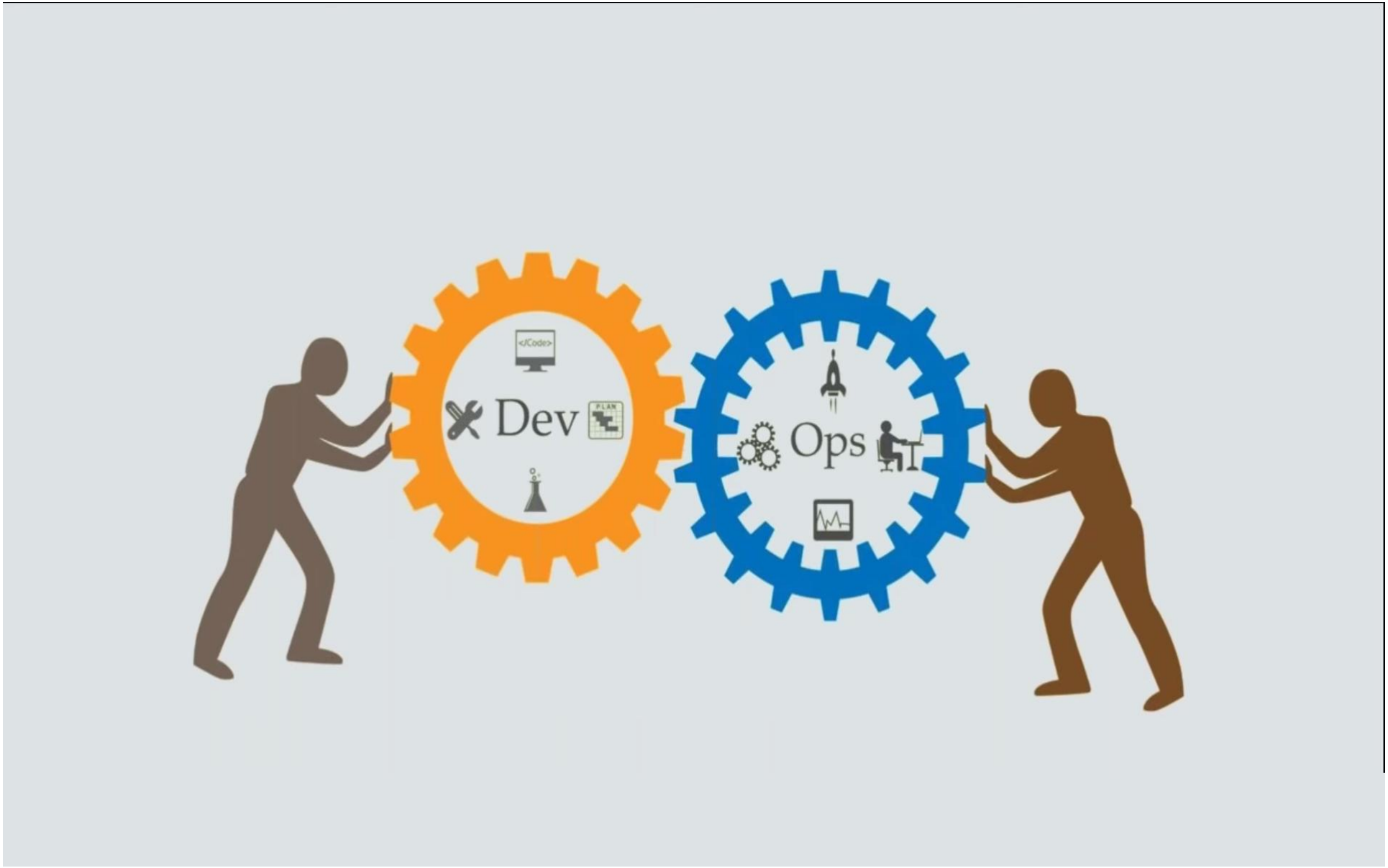
Waterfall

Agile



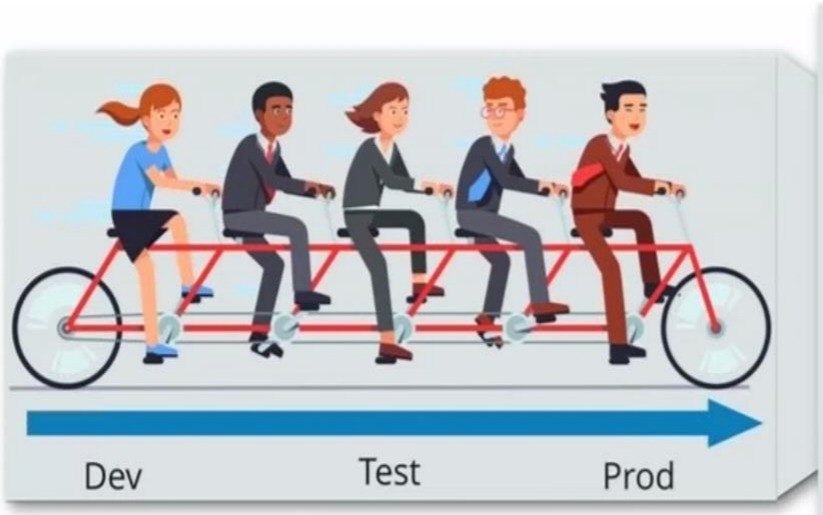
01 POURQUOI DEVOPS?

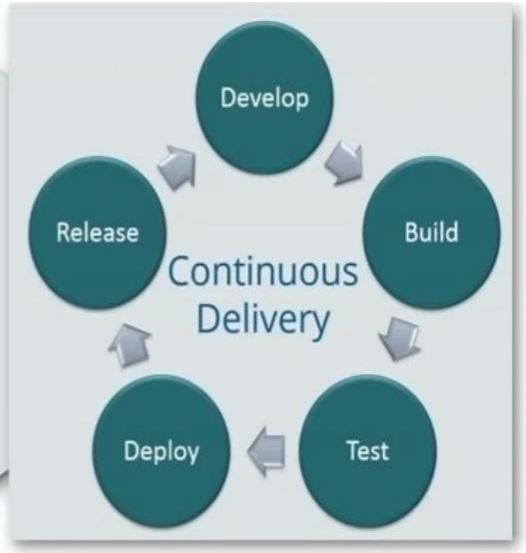



Dev
Ops

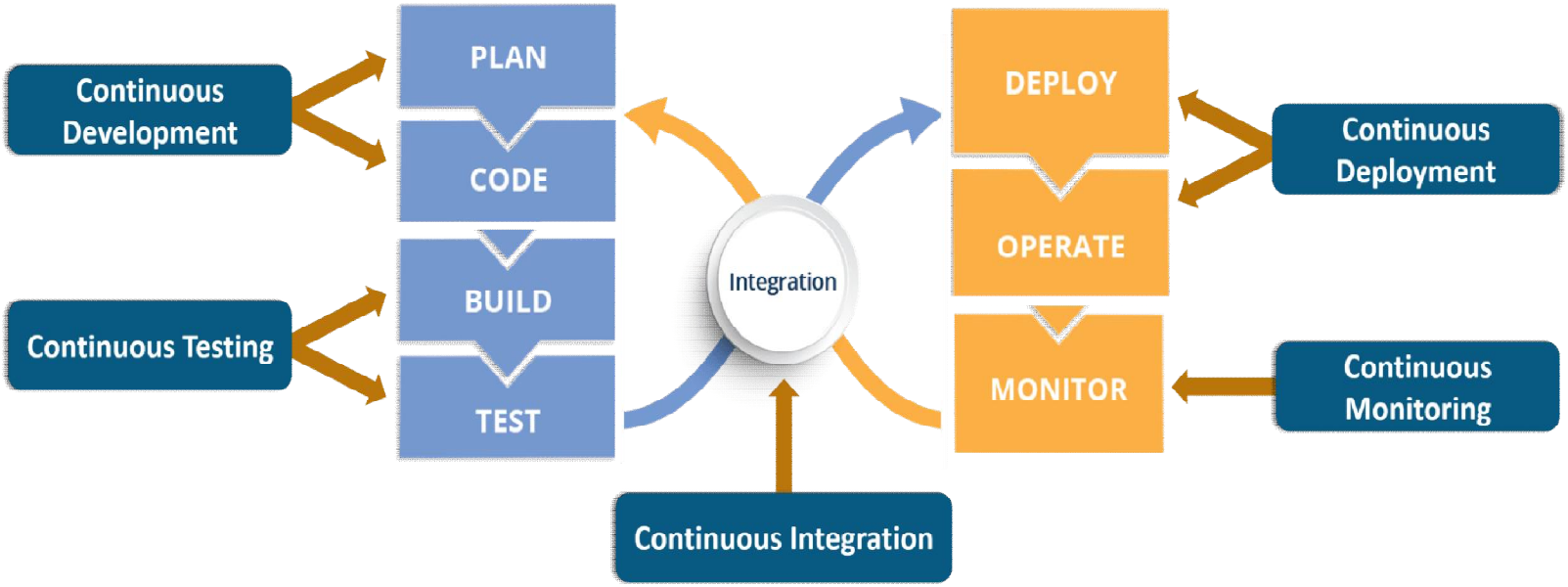
02 RÔLE DU DEVOPS

Le rôle de l’ingénieur DevOps est de rendre l’équipe pas seulement créer, mais aussi maintenir la chaîne d'outils de livraison d'application, de bout en bout

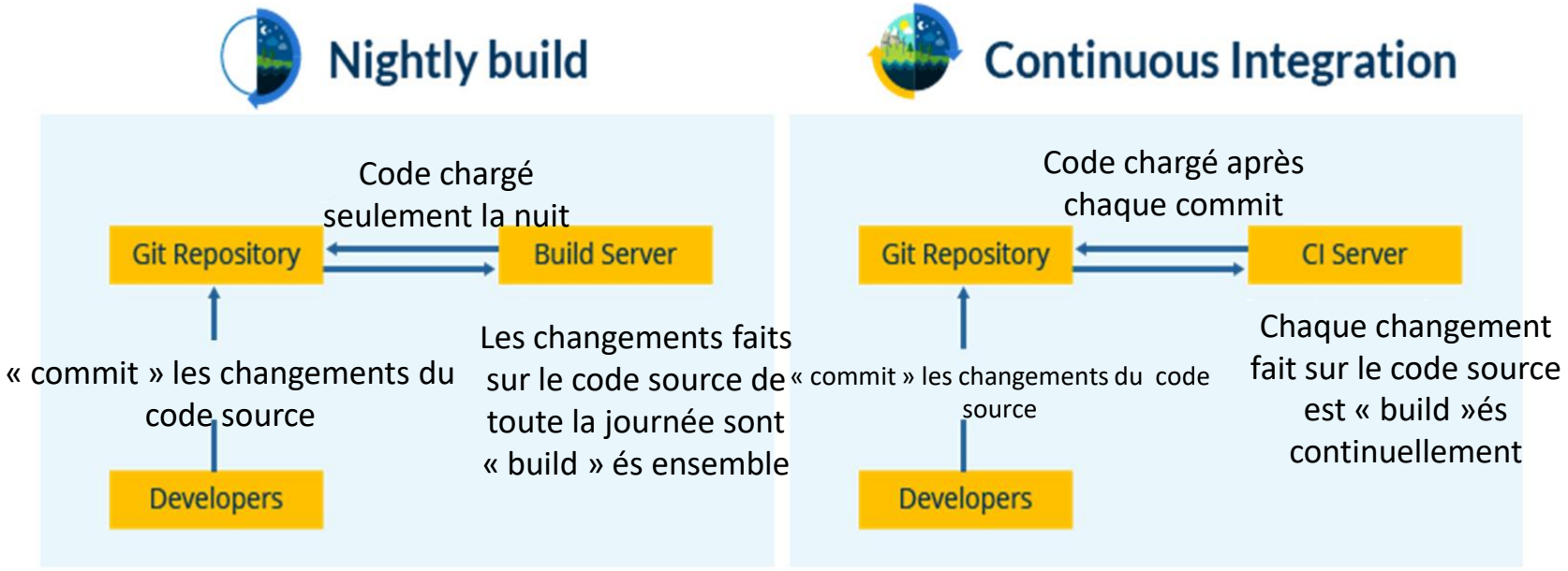




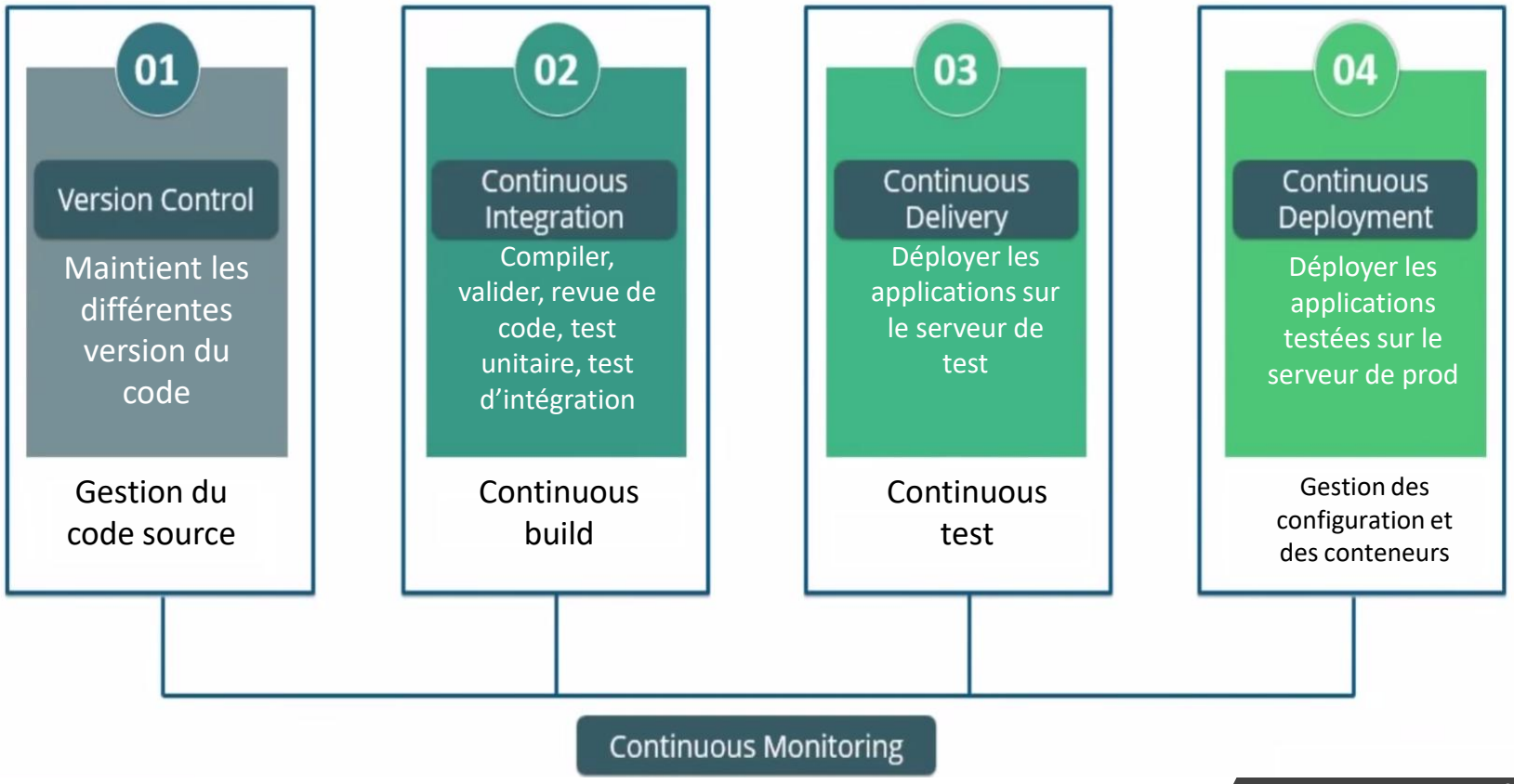
03 LES ÉTAPES DU DEVOPS



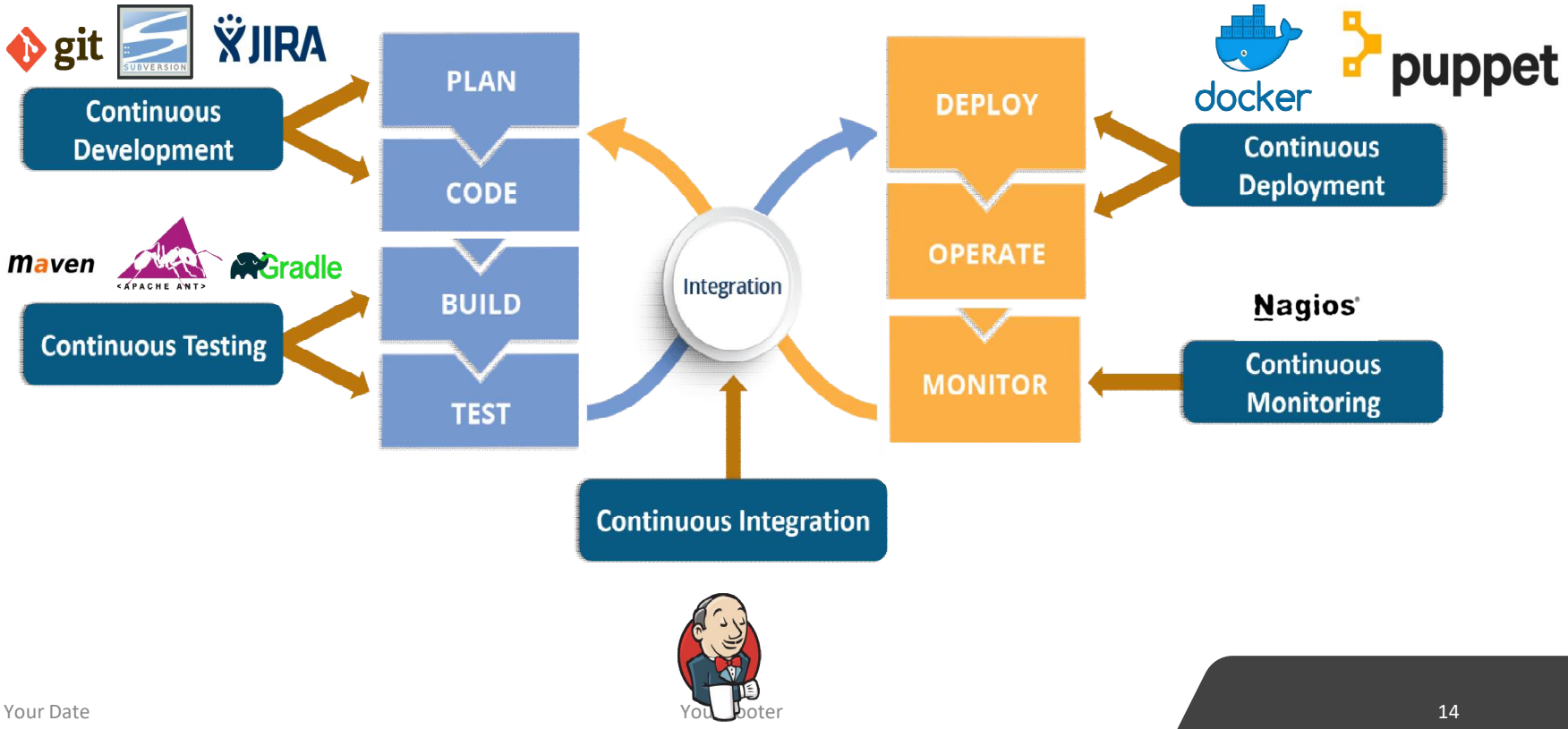
03 POURQUOI « CONTINUOUS »? CAS DE NOKIA



03 LES ÉTAPES DU DEVOPS

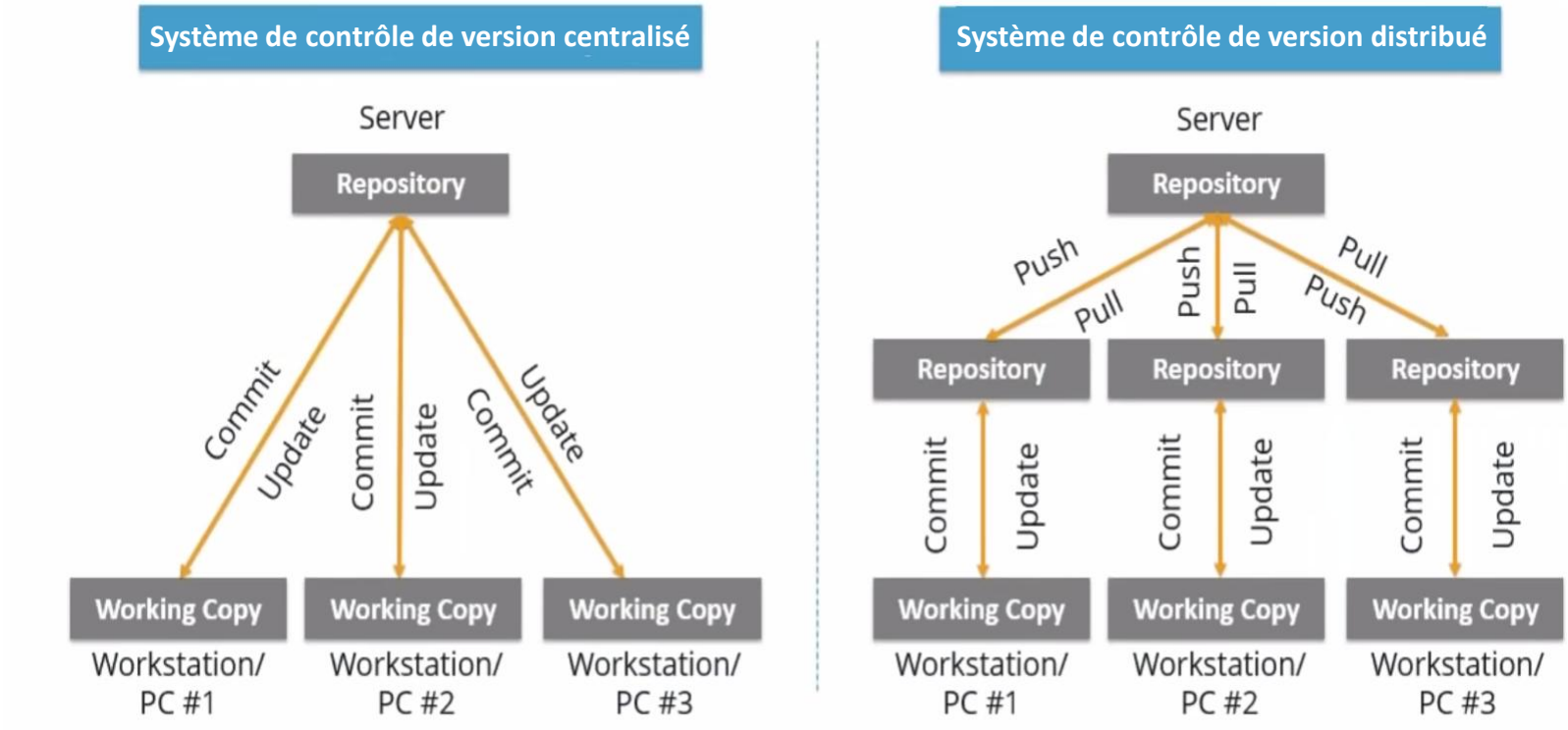


04 LES OUTILS DE DEVOPS



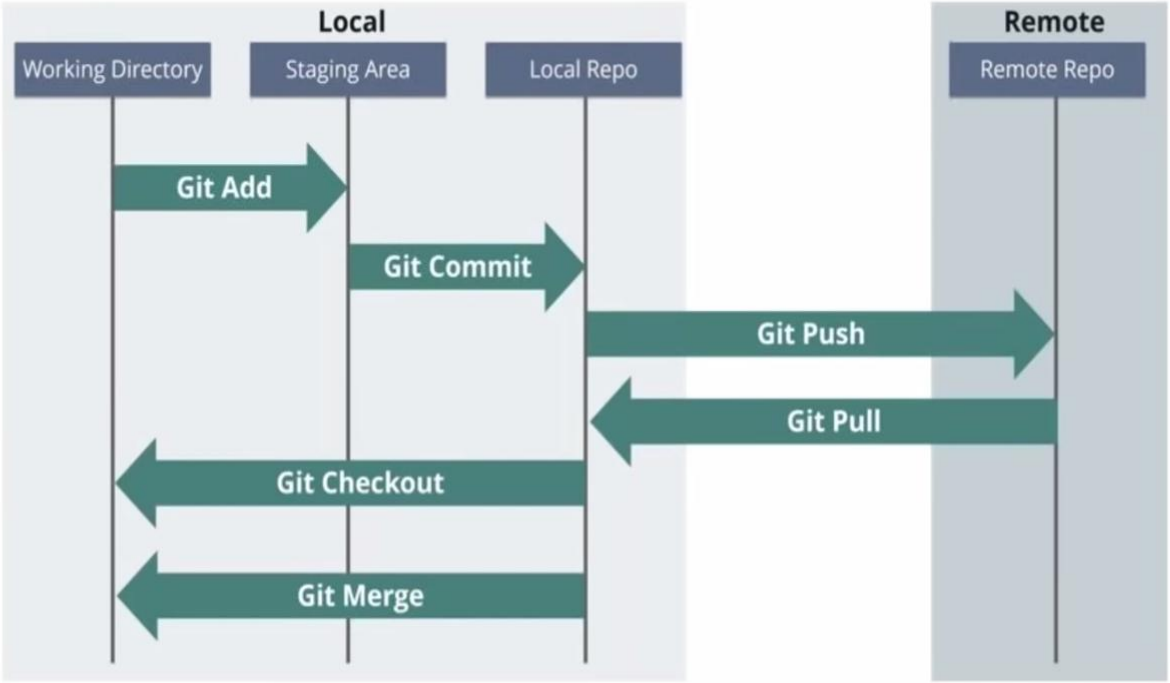
04 LOCAL OU DISTRIBUÉ?

La gestion du changements des documents, du code source etc.

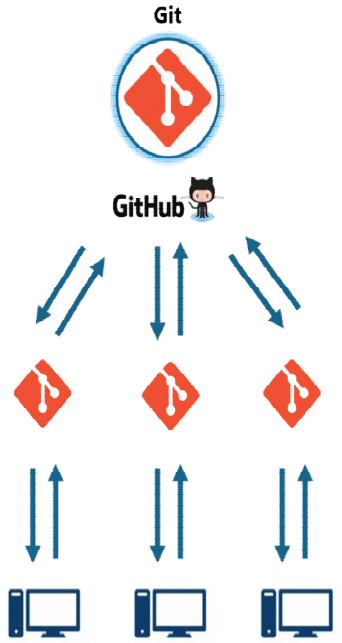


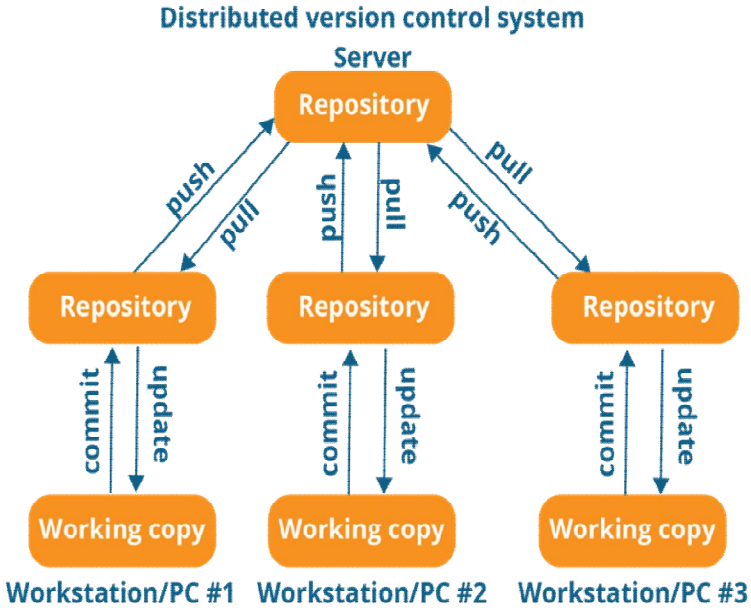
04 GESTION DU CODE SOURCE

Git est un outil de contrôle de version distribué qui supporte des worflows non linéaires assurant la qualité du developpement des logiciels



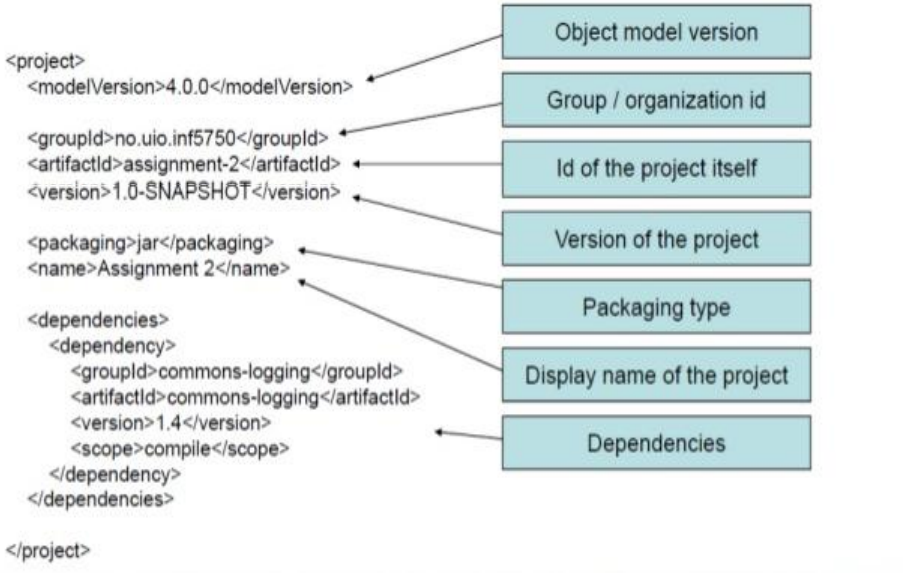
04 GESTION DU CODE SOURCE





04 OUTIL DE BUILD

maven



04 OUTIL DE BUILD

- **Lifecycle**

  Trois built-in lifecycles (aka *build lifecycles*): default, clean, site. (Lifecycle Reference)

- **Phase**

  chaque lifecycle est composé de *phases*, ex. Pour le default lifecycle: compile, test, package, install, etc.

- **Plugin**

  c'est un artifact qui exécute un ou plusieurs goals.

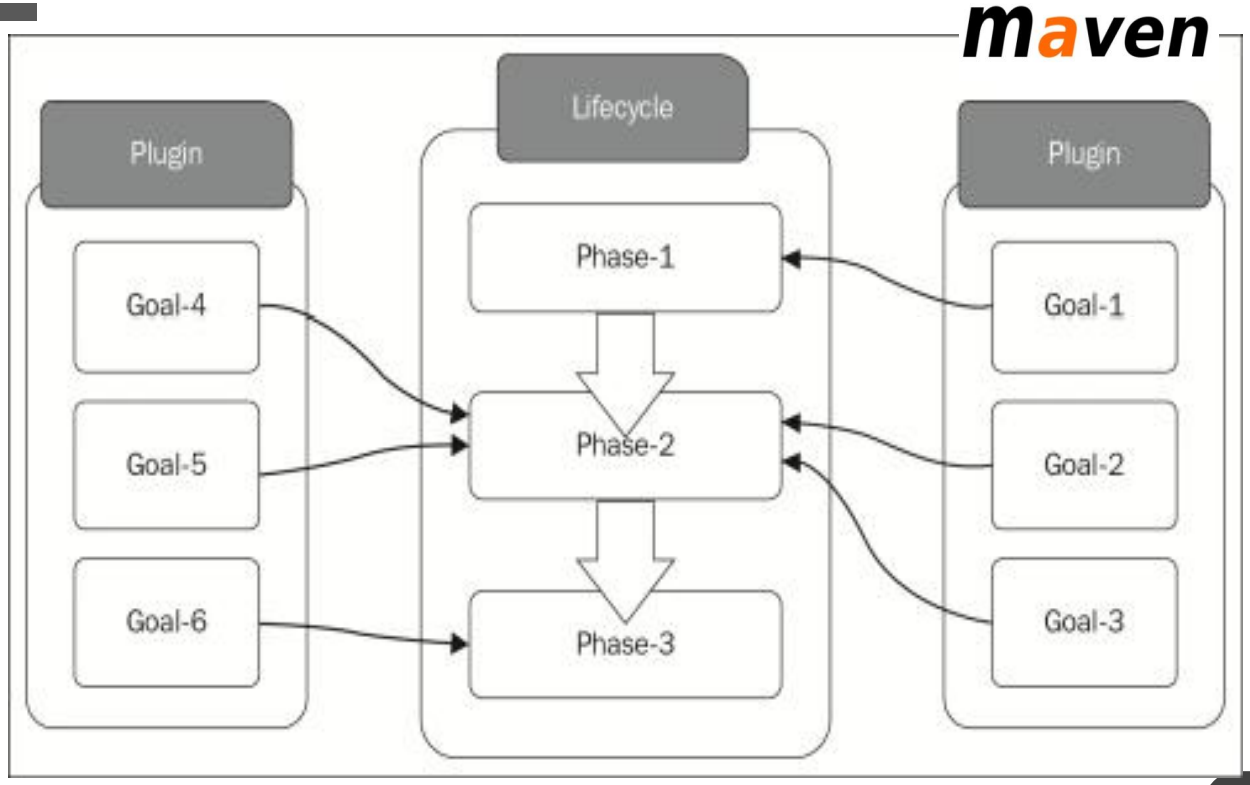
  Selon le type de packaging (jar, war, etc.) plugins' goals sont liés aux phases (Lifecycle Bindings)

- **Goal**

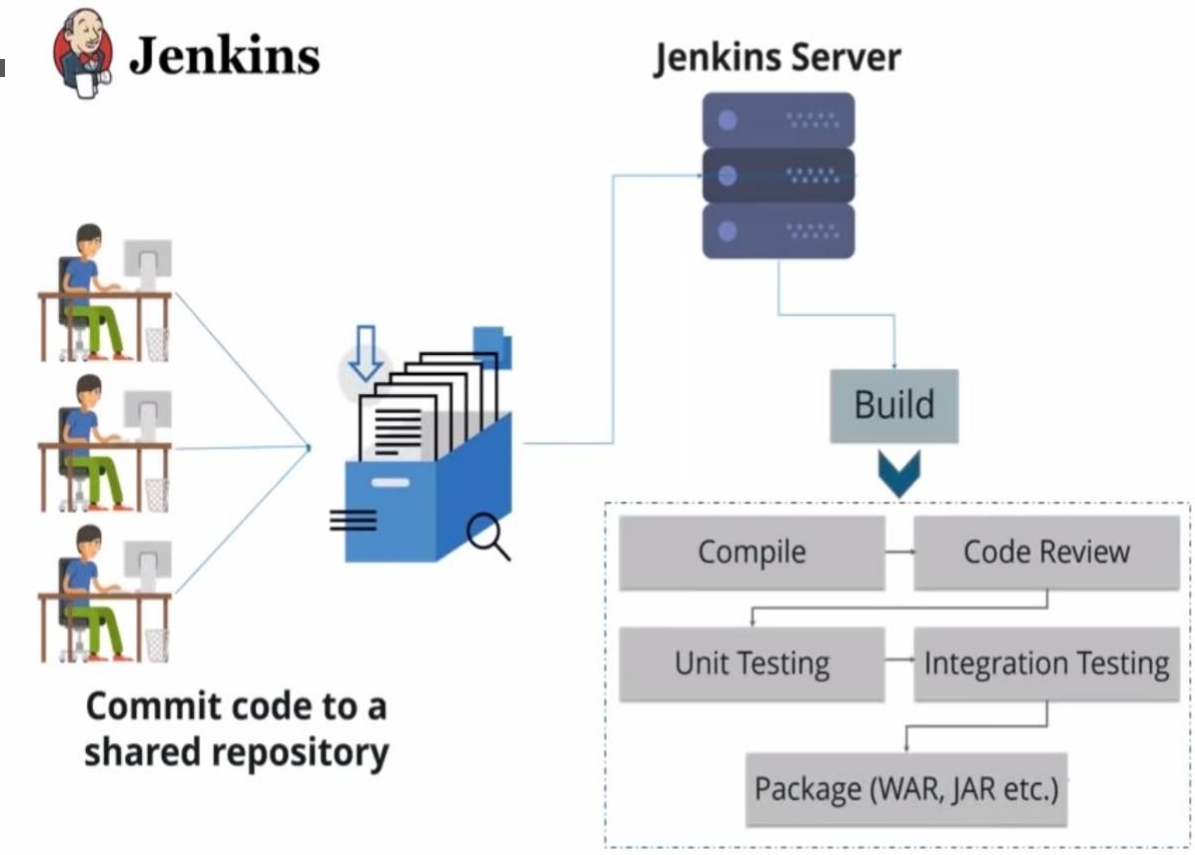
  Une action à s'exécuter.

  On peut ajouter des goals dans le plugin de POM et puis on lie le plugin à une phase

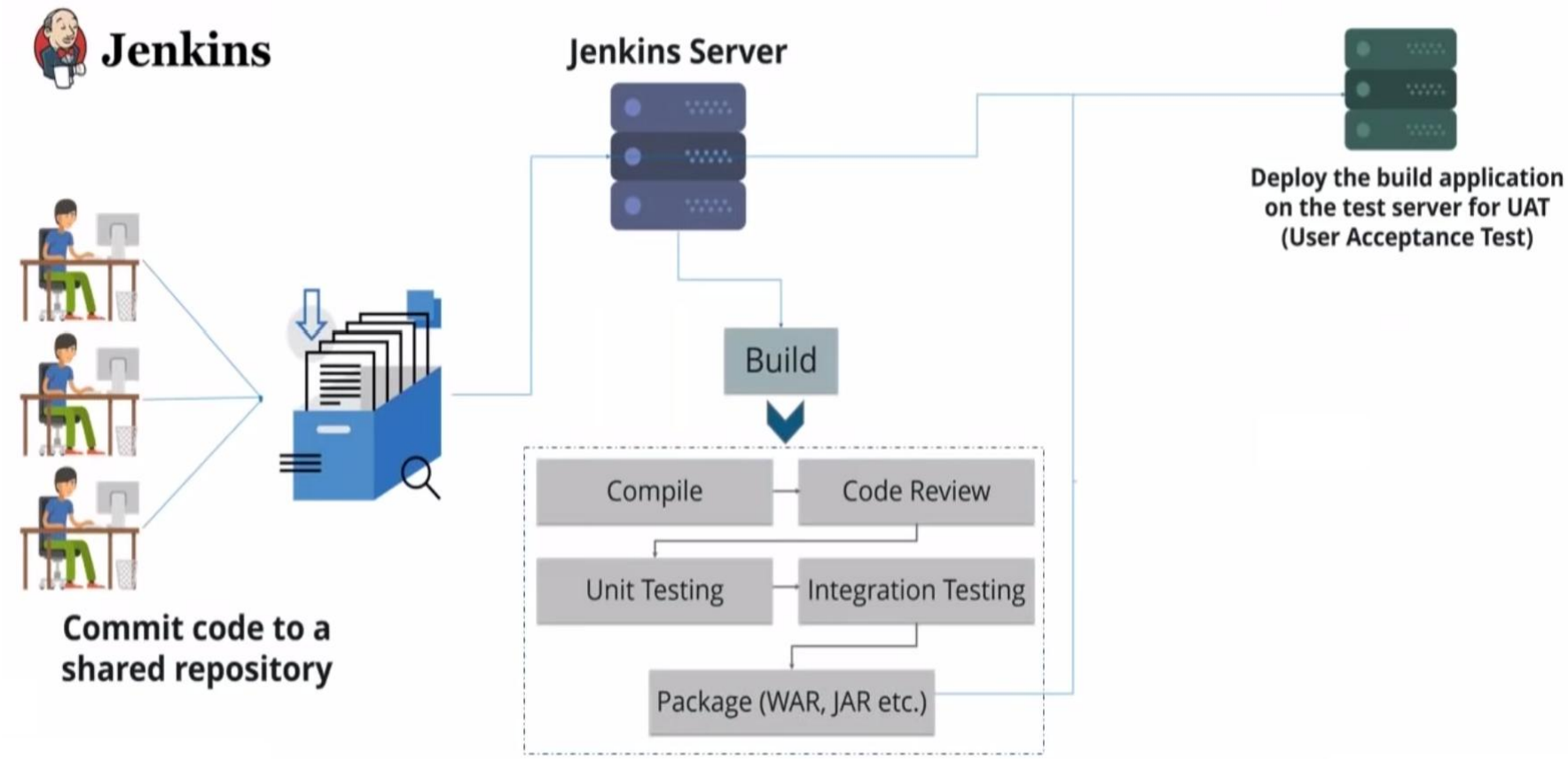
04 OUTIL DE BUILD



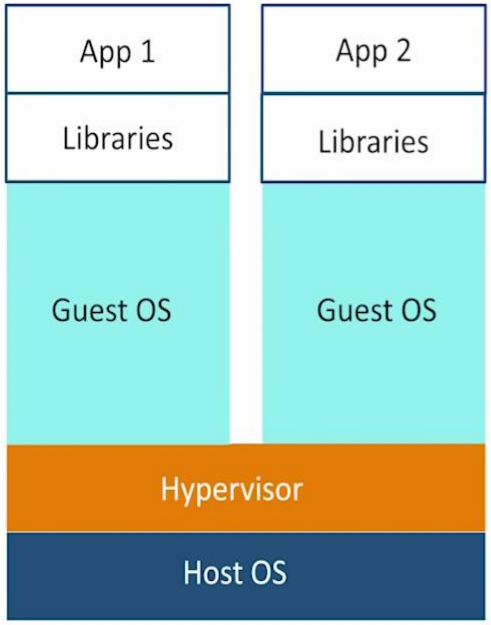
04 CONTINOUS INTEGRATION (CI)

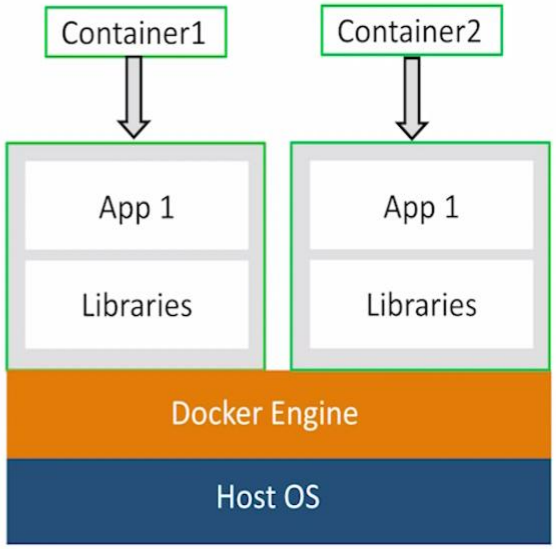


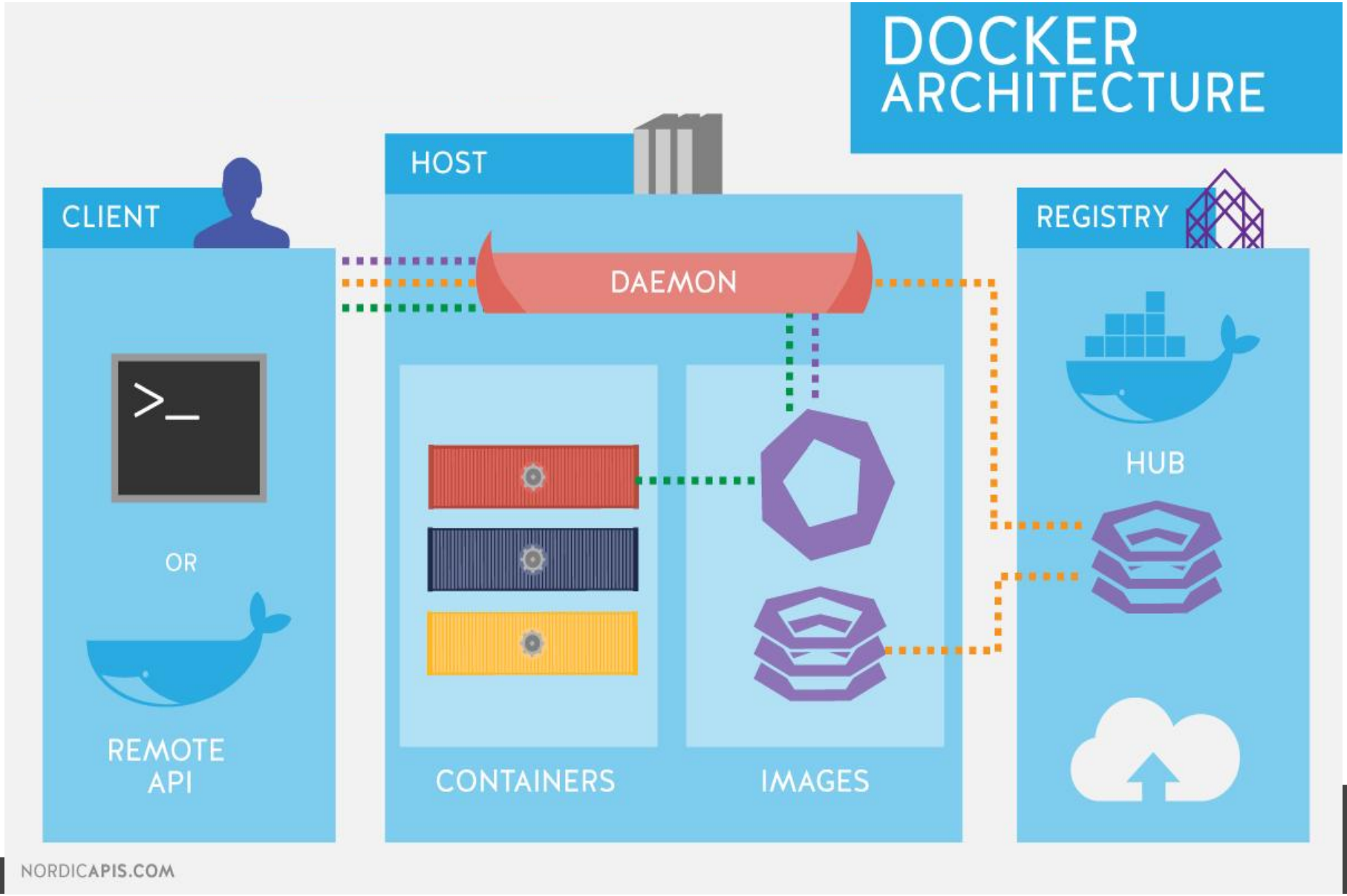
04 CONTINUOUS DELIVERY (CD)



04 CONTINUOUS DELIVERY (CD)

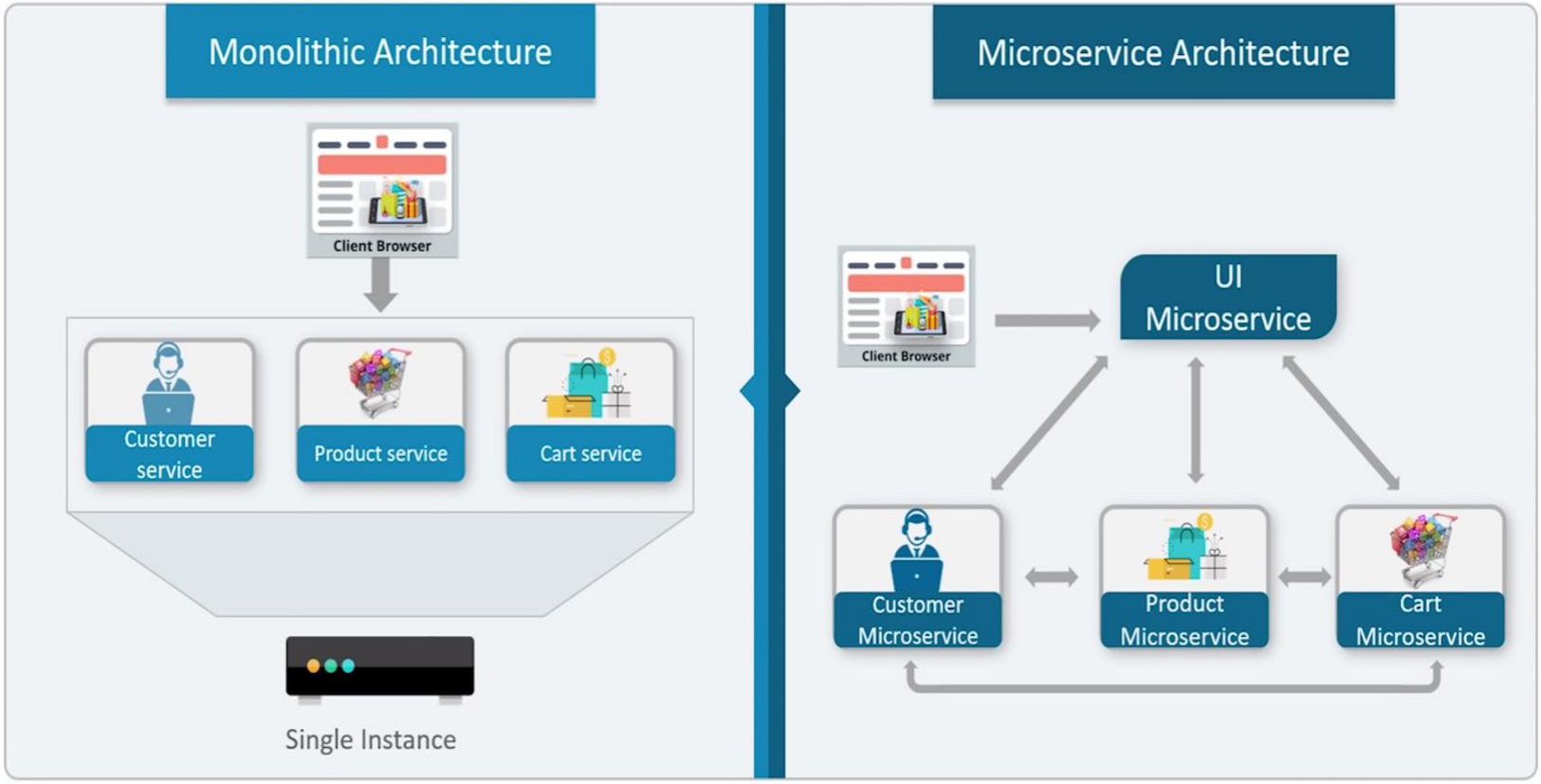





DOCKER
ARCHITECTURE
HOST
CLIENT
REGISTRY
DAEMON
>_
OR
REMOTE
API
CONTAINERS
IMAGES
HUB
NORDICAPIS.COM

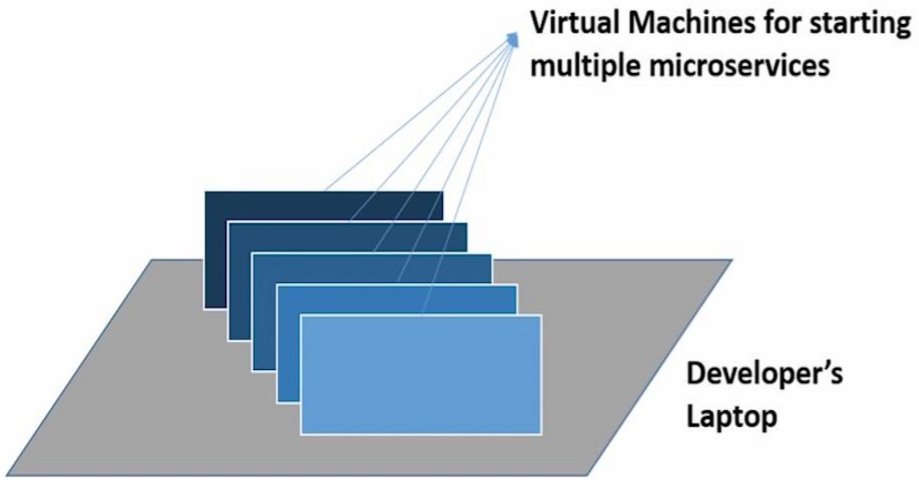
Use Case

05 ARCHITECTURE ORIENTÉE SERVICES

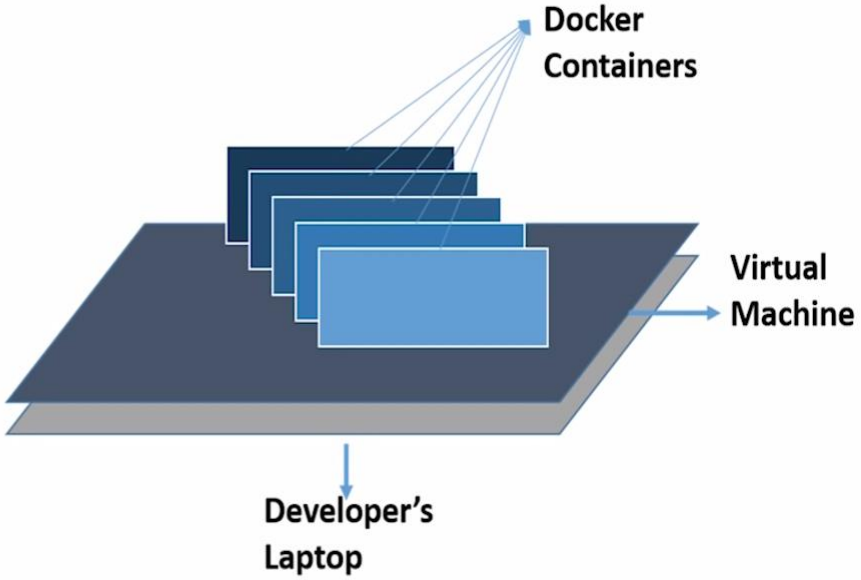


05 VIRTUALISATION VS DOCKERISATION

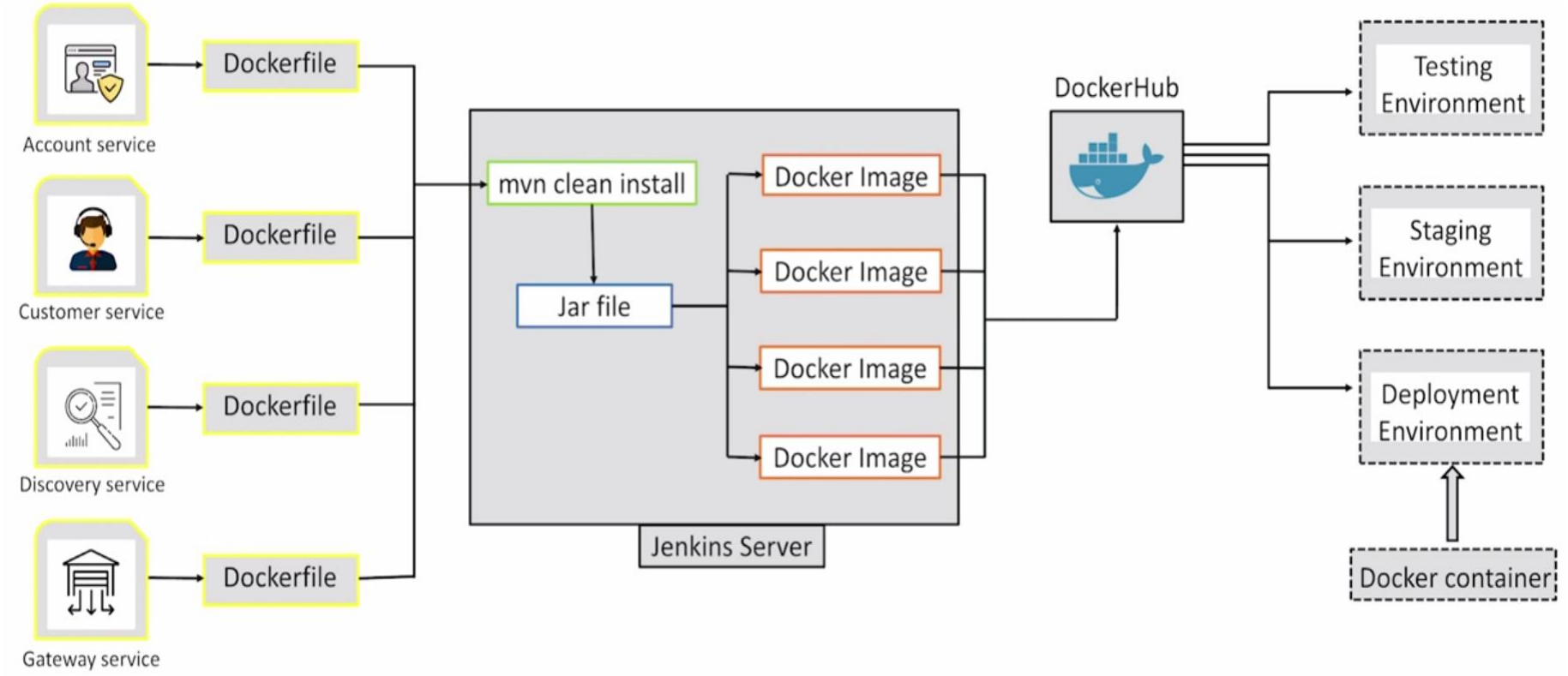
Développer une application nécessite beaucoup de microservices dans une seule machine. Ainsi, si vous lancez cinq de ces servives, vous avez besoin de cinq VMs sur votre machine.



Vous lancez une seule VM en lançant plusieurs conteneurs Docker pour chaque microservices



05 CI CD PIPELINE



Hands On

06 HANDS-ON

- Examiner un code avec maven
- Push avec git vers github
- Configurer jenkins
- Faire un build (CI avec jenkins)

07 A FAIRE POUR LA PROCHAINE FOIS

- Hands-On
  - Un CR sur les étapes vues en classe
- CI CD pipeline avec jenkins et docker
  - Un compte rendu expliquant les différentes étapes de la pipeline
  - Une implémentation de la pipeline