Génie logiciel

Déploiement

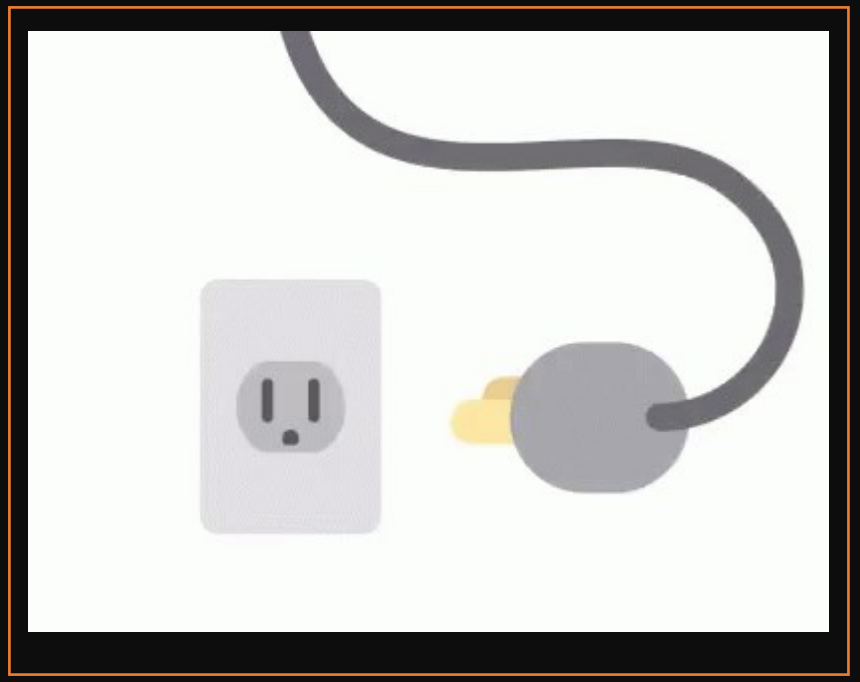
Louis-Edouard LAFONTANT

geodes

Université de Montréal

Déploiement

Il est temps de livrer la marchandise!



ATTENTION ⚠

- Le produit est très fragile (nature du logiciel).
- Le produit peut changer, il faut penser à l'après!

➢**Il faut une stratégie de déploiement.**

Versionnement d’un logiciel

Pré-alpha

- Encore en développement, avant les tests formels complétés

Alpha

- En mode tests fonctionnels

Beta

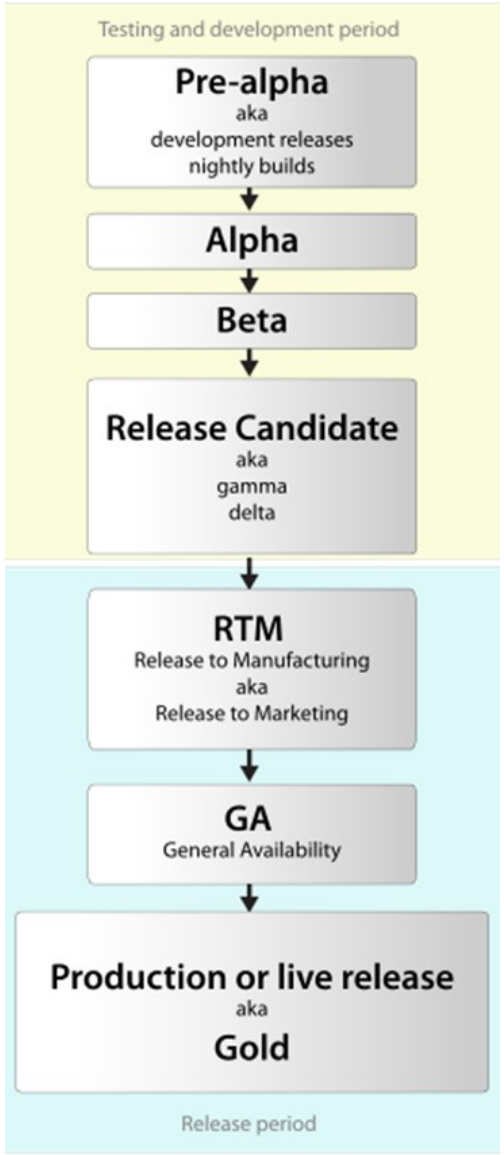
- logiciel fonctionnel, testé, dans les mains d’utilisateurs contrôlés

Candidat à la livraison

- Potentiellement la version finale à moins de découverte d’autres défauts
- code source est figé

En production

- Disponible au public
- version commercialisée



Phases d'un logiciel

Phase de développement

Development releases
Nightly builds

Pre-alpha
En développement
Avant tests formels

→

Alpha
Tests fonctionnels
Peut avoir des fautes majeures

Feature complete

→

Preview release

Beta
Test d'utilisabilité
Peut avoir des fautes mineures

Code complete

→

Release Candidate

RC
Produit potentiel
Documentation
Correction mineure

Version "*stable*"

RTM
Prêt à être distribué
Denier build

Product ready

→

GA
Commercialisation
Mise en vente

Product available

→

Production
Disponible au public
Version commercialisée

Release to Manufacturing
Release to Marketing

General Availability

Live release

Phase de publication

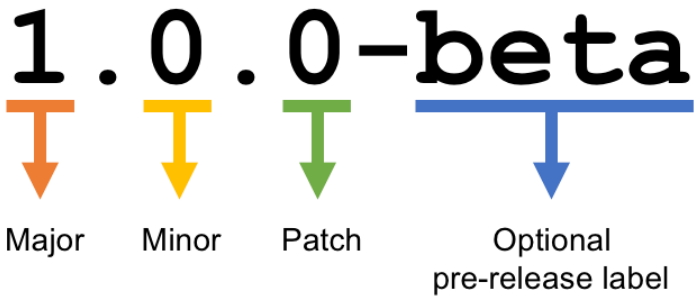
Séquence de version typique

Deux premiers nombres marquent la version (p.ex. 1.3)

Marquent le statut (Ex: α, β, RC)

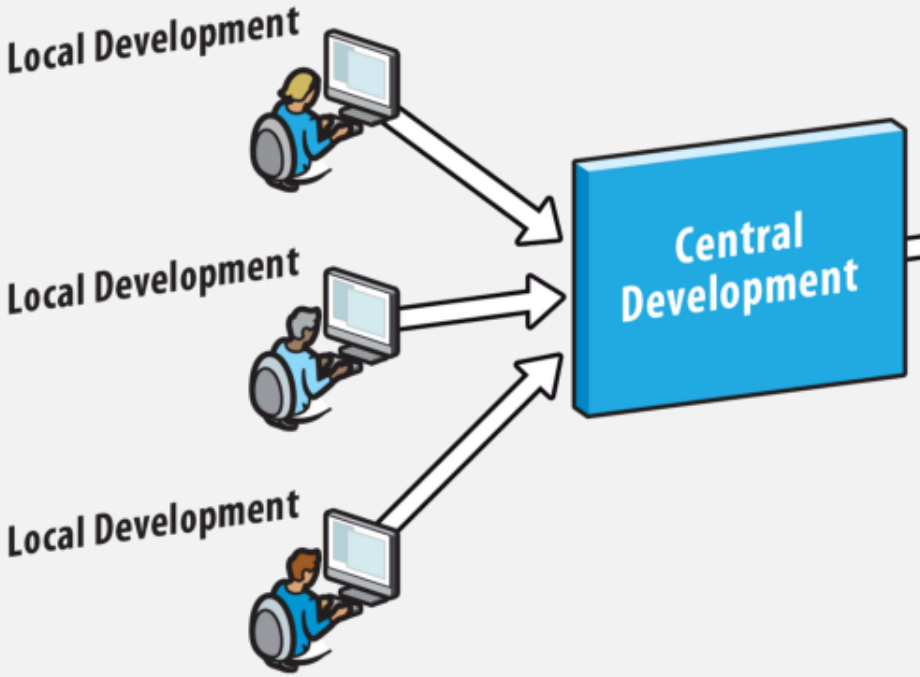
```
[0-9]+ · [0-9]+ · ([0-9]+ ·)? [0-3] · [0-9]+
```

- Version 1.2 contient des améliorations mineures par rapport à Version 1.1
- Version 2.0 contient des améliorations majeures par rapport à Version 1.9
- 1.2.0.1: alpha 1 de la version 1.2
- 1.2.1.2: beta 2 de la version 1.2
- 1.2.2.3: release candidate 3 de la version 1.2
- 1.2.3.0: version 1.2 en production



[Semantic Versioning (semver.org)]

Environnements



Environnement de développement

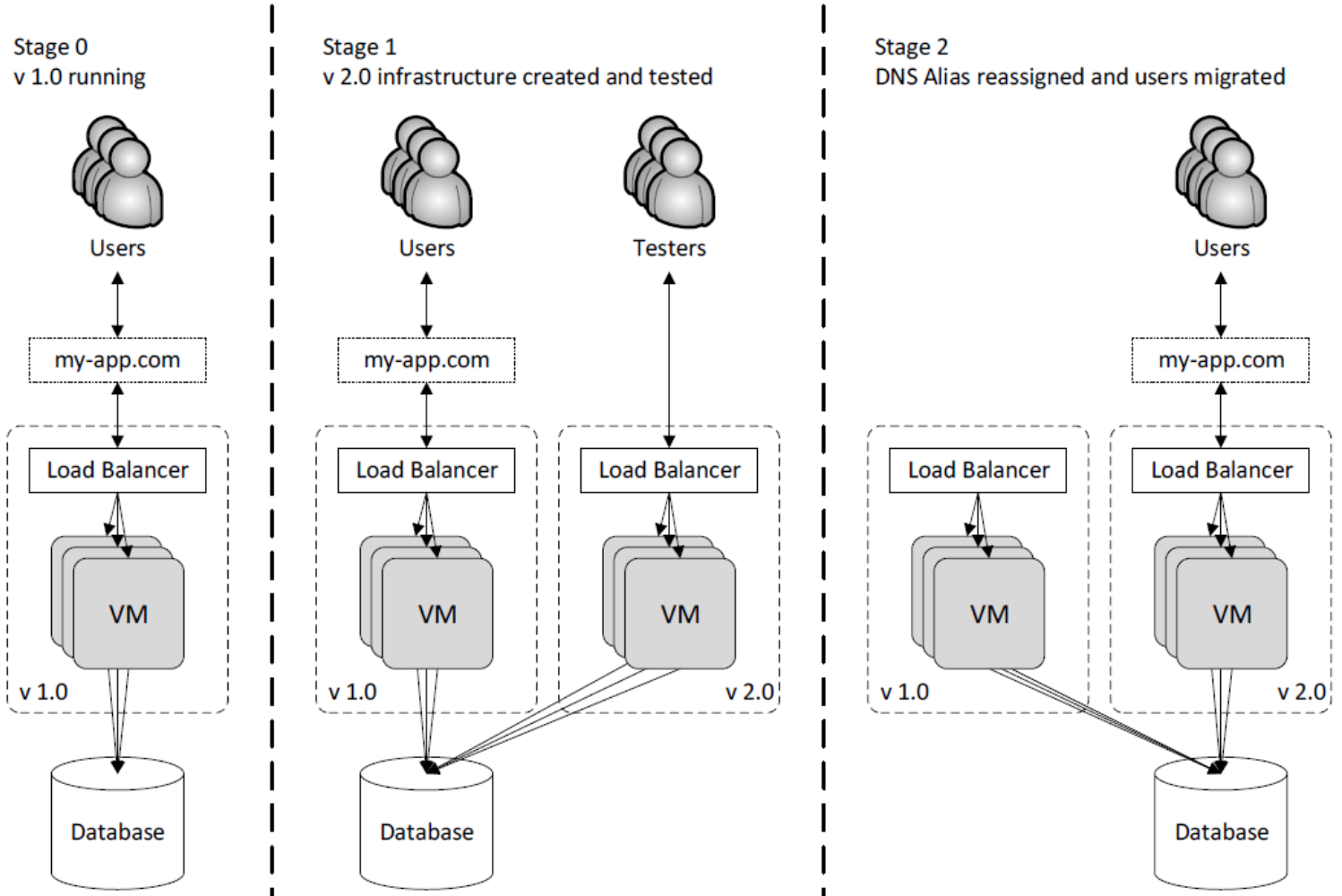
- Ordinateur personnel du développeur contenant tout le nécessaire pour le développement
  - IDE, Base de données et serveur web locaux

Environnement de test (pré-production)

- Simule l'environnement de production
- Roule souvent sur une **machine virtuelle**
- Peut avoir un système d'exploitation différent de l'environnement de dev

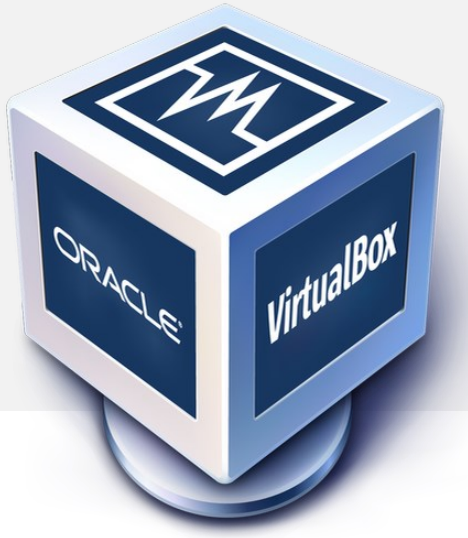
Environnement de production

- Vrai environnement où le logiciel est installé
- Utilisateurs finaux opèrent dessus ou communiquent avec



Environnements et machines virtuelles

Machine virtuelle



- Abstraction complète de la machine
- Interagit avec les composantes physiques de l'ordinateur via *hypervisors*

✓**Émuler complètement plusieurs machines**

- ➢ Plusieurs OS
- ➢ Plusieurs infrastructures serveurs.

× **Utilise beaucoup de ressources**

vmware®

Containers

- Paquets contenant toutes les composantes (services, librairies) pour exécuter une application
- Utilise les fonctionnalités de l'OS hote pour en simuler un autre

✓ **Solution légère**

- ➢ N'émule pas tout le système
- ➢ Utilisation efficiente des ressources

✓ **Favorise l'architecture de microservice**

× **Écosystème fracturé**

× **Persistence des données peut être compliqués**

Virtualisation

Machine virtuelle

Abstraction complète de la machine

Interagit avec **composantes physiques de l'ordinateur** via *hyperviseurs*

✓**Émuler complètement plusieurs machines**

- ➢Plusieurs OS
- ➢Plusieurs infrastructures serveurs.

×**Utilise beaucoup de ressources**

vmware®

Conteneur

Contient toutes les dépendances (services, lib…) pour exécuter une app

Utilise les **fonctionnalités de l'OS hôte**

✓**Solution légère**

- ➢N'émule pas tout le système
- ➢Utilisation efficiente des ressources

✓**Favorise les microservices**

×**Écosystème fracturé**

×**Persistence des données**

docker

kubernetes

Déploiement

Transition de l’application d’un environnement à un autre

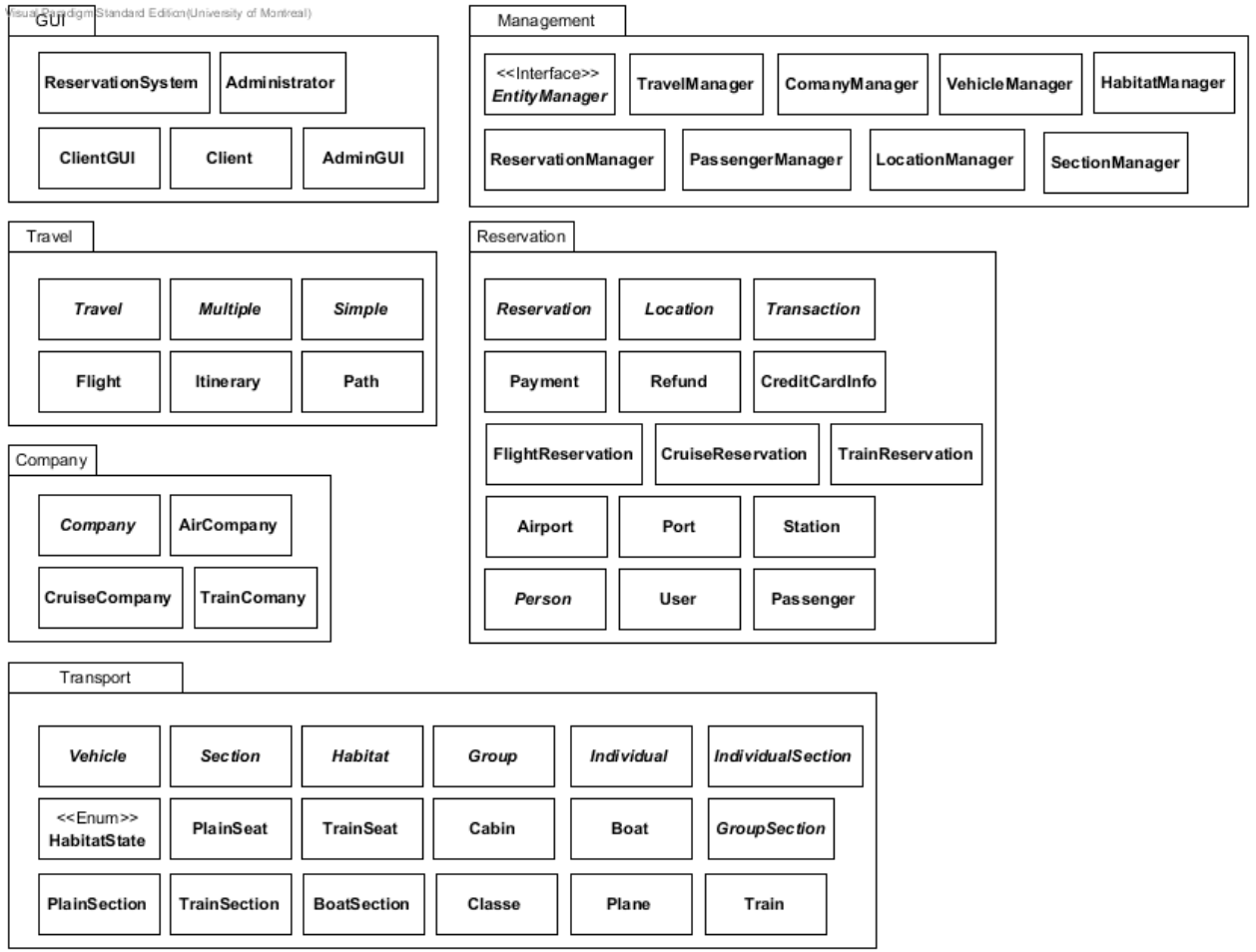
- Consiste en différentes activités
  - Assemblage et configuration
  - Livraison
  - Transfert
  - Installation
  - Activation
  - Mise à jour

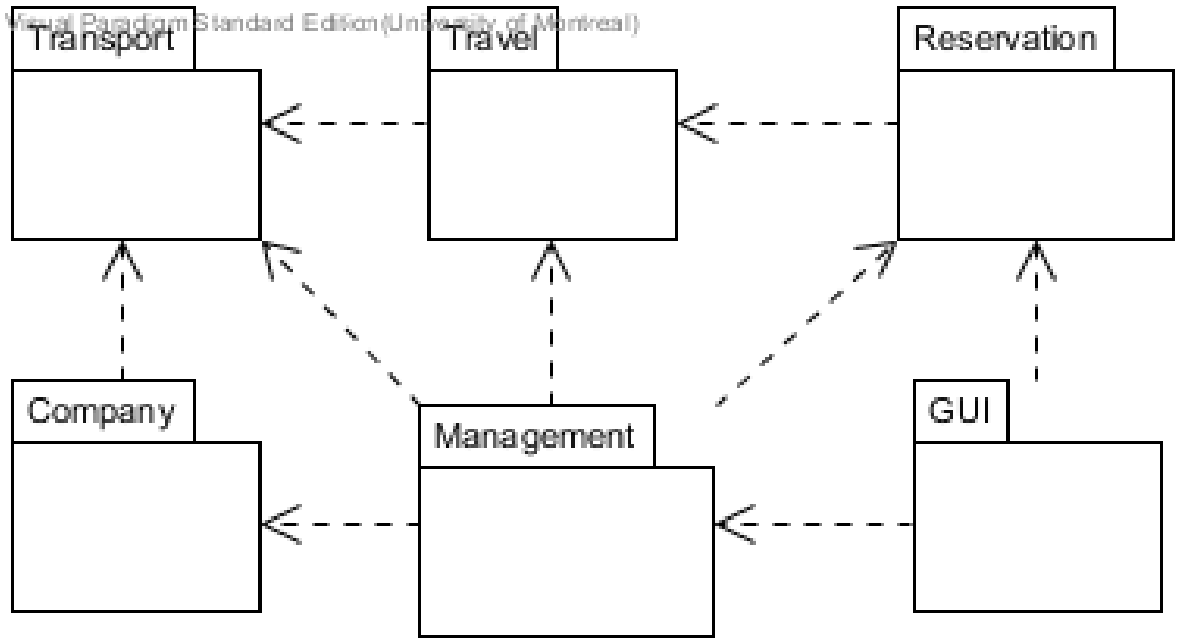
Assemblage et configuration

- Chaque **composant** définit les interfaces qu'elle fournit et qu'elle requiert
- Les composants interdépendants sont assemblés dans un **artéfact *(assembly)*** qui peut aussi requérir des interfaces

  ⇒Déployer une application revient à **instancier** les *assemblies* dans un env.
- Chaque composant est libellé par une **version**
  - Ordre de sa révision
  - variante spécifique à l'environnement ciblé

Diagramme de paquets





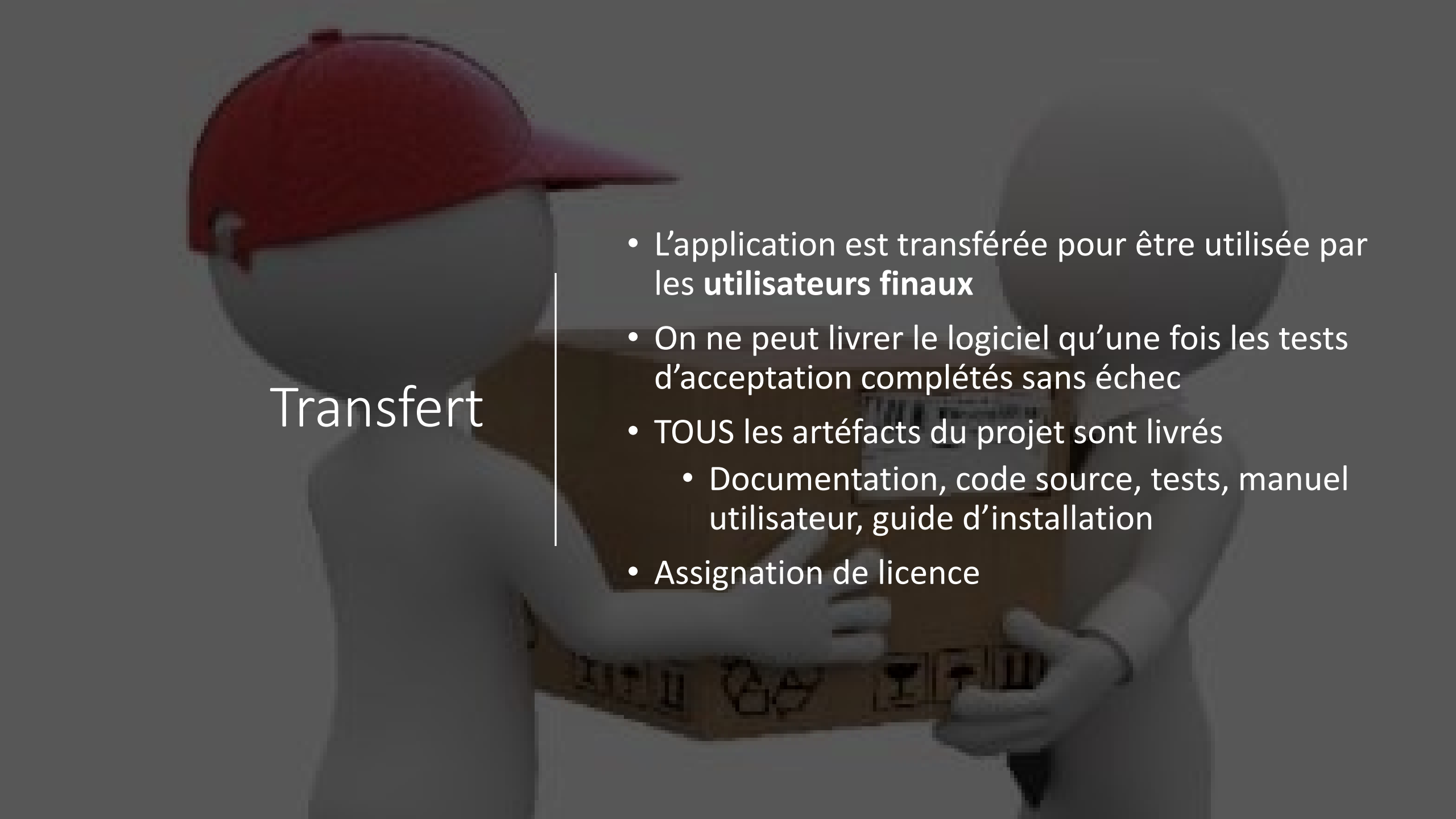
Livraison

- On forme un paquet qui contient les *assemblies* et *metadatas* de son contenu
  - Version, auteur, nom du produit, producteur, etc.
- Paquet peut contenir plusieurs implémentations des composants
  - satisfaire les besoins de différents environnements matériel et logiciel

Unix: RPM, tar.gz

Windows: MSI, DLL

Android: APK

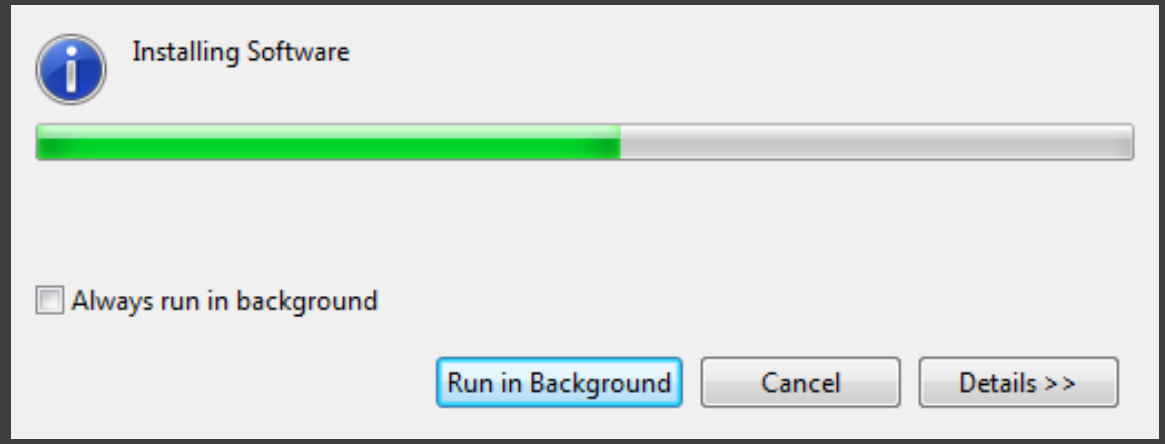


Transfert

- L’application est transférée pour être utilisée par les **utilisateurs finaux**
- On ne peut livrer le logiciel qu’une fois les tests d’acceptation complétés sans échec
- TOUS les artéfacts du projet sont livrés
  - Documentation, code source, tests, manuel utilisateur, guide d’installation
- Assignation de licence

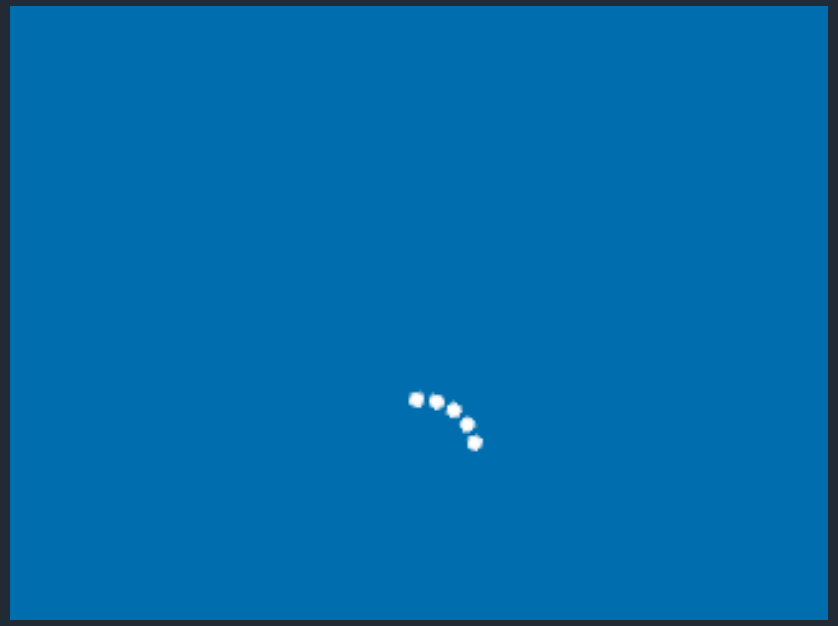
Installation

- **Déballe** le paquet *(unpacking)*
- Vérifie que les exigences du système hôte sont satisfaites
- Crée (ou copie) les fichiers nécessaires pour exécuter l'application sur le disque
- Ajoute des données de **configurations**: registres, variables d'environnement, fichier de configuration
- Déclenche **l'activation** du logiciel



Mise à jour

- Change une partie du logiciel installé à cause de la livraison d’une nouvelle version
- Peut nécessiter la désactivation ou la désinstallation du logiciel pour effectuer les changements nécessaires
- Logiciel peut se mettre à jour sans intervention humaine
  - Vérifie automatiquement si une nouvelle version est disponible
  - Bonne pratique pour un logiciel sur le serveur, mais pas pour un logiciel sur le client !



Enjeux de la livraison

Processus d'affaires
- Changement dans la façon de travailler
- Est-ce que le logiciel répond vraiment aux besoins du client ?

Formation
- S'assurer que les utilisateurs savent utiliser adéquatement le logiciel
- Utilisateur expert

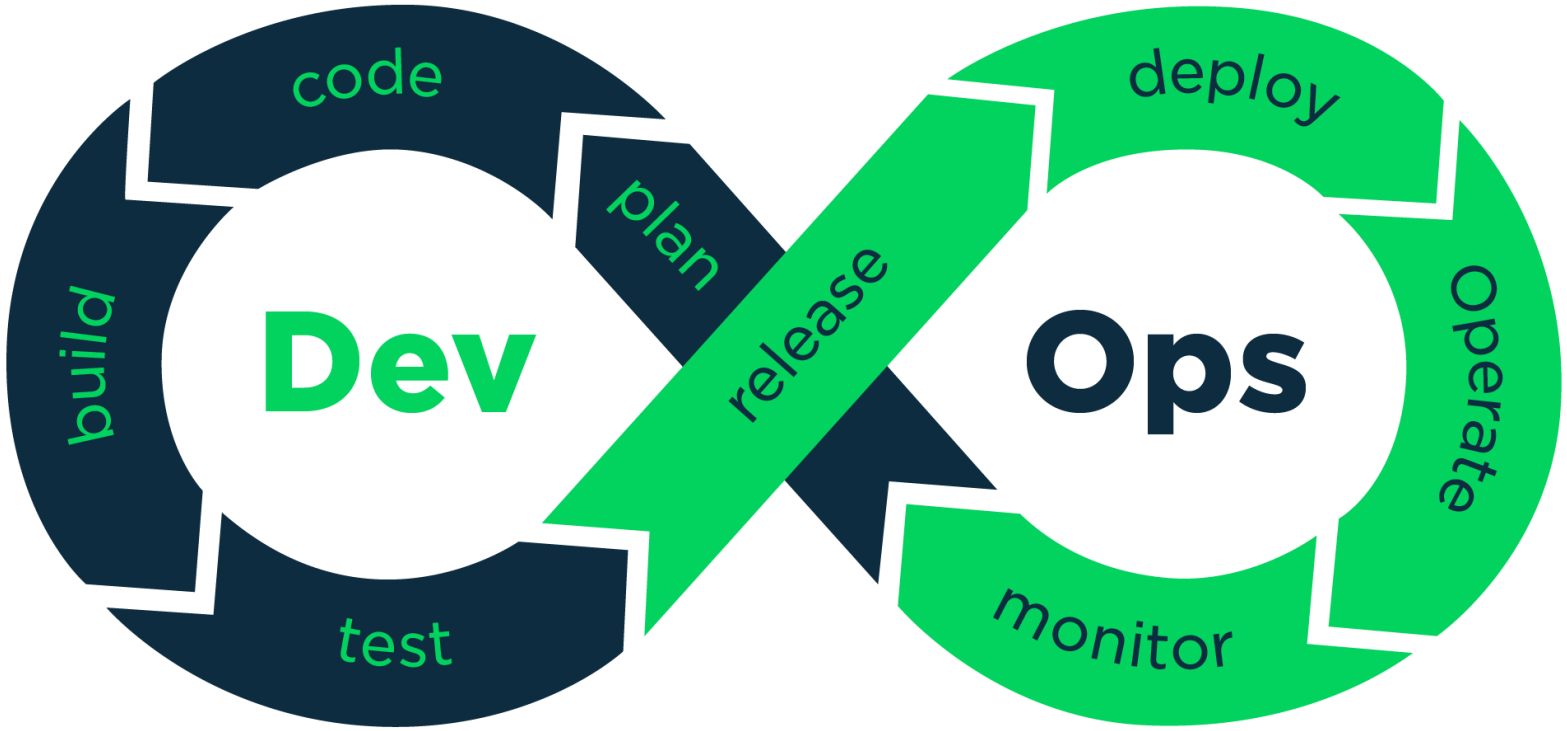
Stratégie de déploiement
- Comment installer le logiciel ? Intervention humaine ?
- Requiert un administrateur système ? Comment le rendre disponible ?

Équipement
- S'assurer que le matériel de l'utilisateur satisfait les prérequis de l'app.

Intégration
- Comment le logiciel s'intègre-t-il avec les autres logiciels du client ?

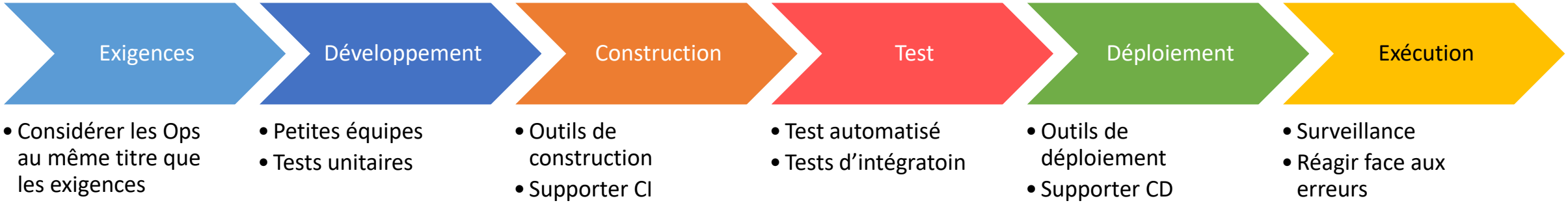
DevOps

Développeurs et opérateurs

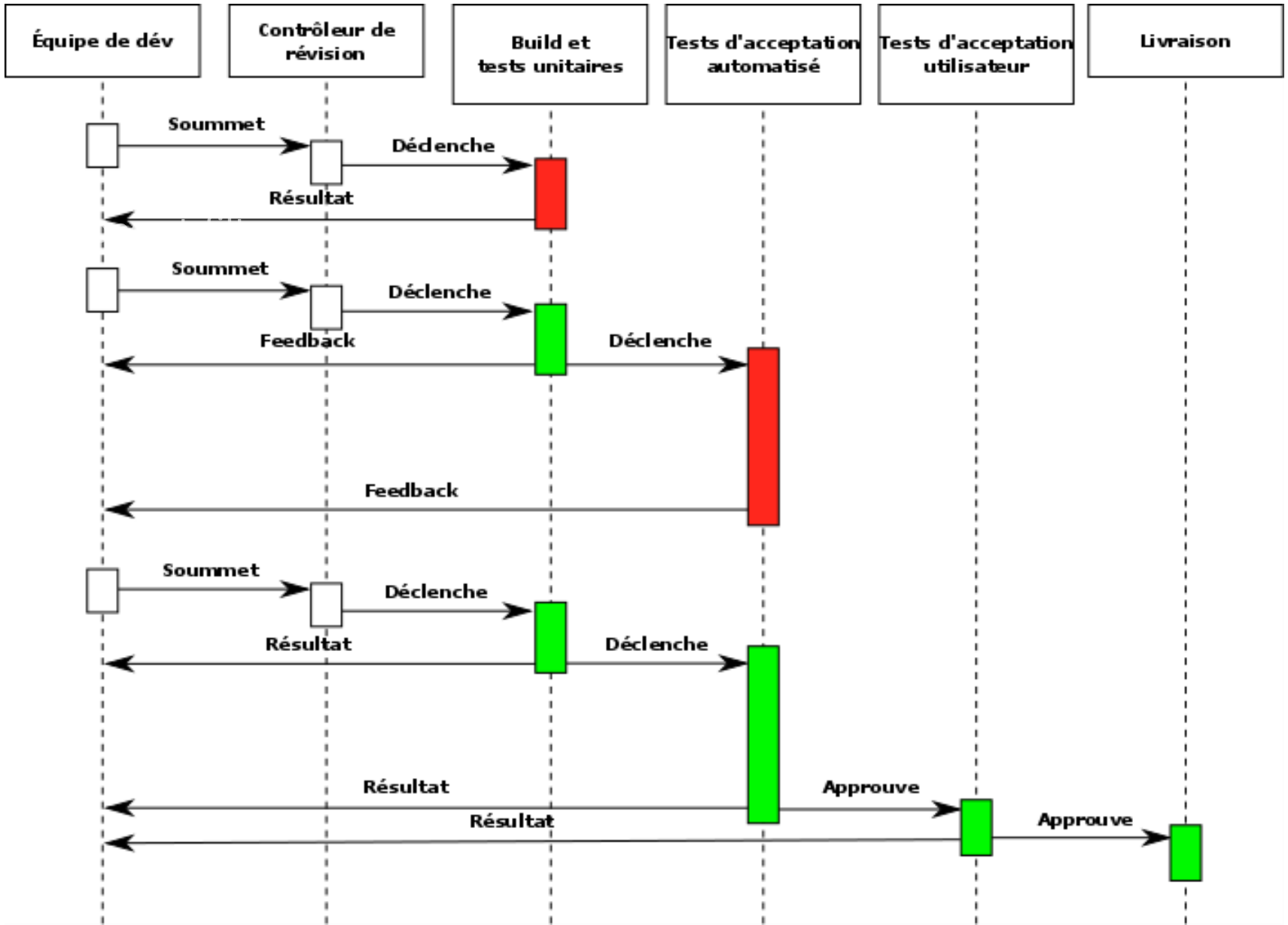


Pratique DevOps

- Inclure les Ops tout au long du processus de développement
- Déployer en continue, tout automatiser
- Instaurer un processus de déploiement suivi par tous
- Développement de l'infrastructure doit suivre les mêmes pratiques que le code de l'application
  - Ops utilisent souvent des scripts ad hoc



Flux de travail (automatisé)



Intégration continue & Livraison continue (CI/CD)

- Intégration continue
  - Automatisation de l’intégration des changements apportés au code.
  - Des tests automatisés sont utilisés pour éviter l'introduction de fautes (bugs) ou d'une incompatibilité entre le nouveau code (branche) et le code principal (main).
- Livraison continue
  - Infrastructure d'automatisation du déploiement/publication ciblant un ou plusieurs environnements

Jenkins

circleci

Azure Pipelines

TeamCity

Outils de compilation (*build*)

Outil de permettant d'automatiser diverses opérations (répétitives) du développement logiciel (ex: compilation, documentation, tests…)

- ✓Si plusieurs versions sont souhaitées
  - ✓Différentes versions ou architectures ciblé
  - ✓Changer entre les versions de débogage et release
- ✓Accélère le processus de construction
  - ✓Exécuter automatiquement (ou sauter) des cas de test
  - ✓Automatisation des sauvegardes / archivage

Outils de configuration

Outils permettant d'automatiser le déploiement du logiciel et la configuration des paramètres et variables pour l'environnement physique et virtuel.

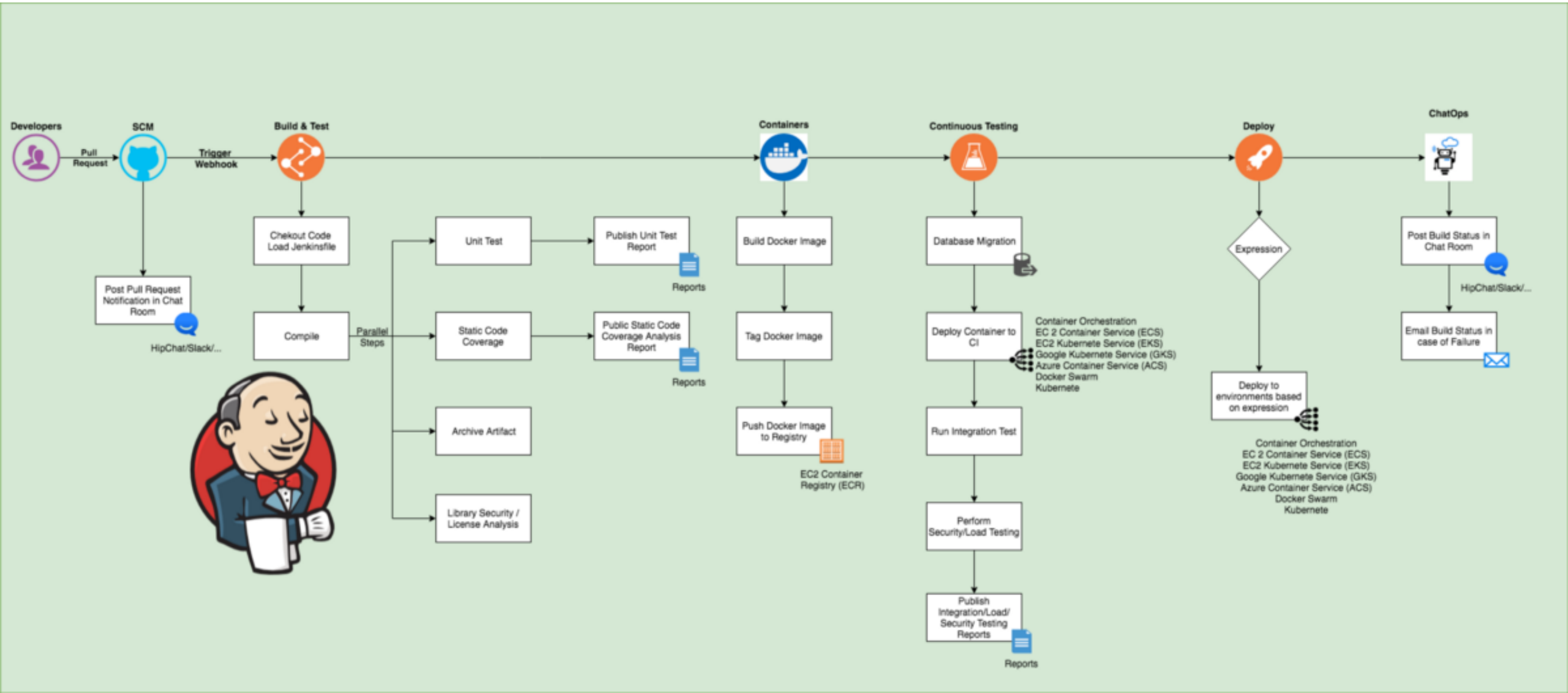
- ✓Maximise l'efficacité du serveur (allocation des ressources)
- ✓Garantis l'application de normes dans le code
- ✓Indépodence: Même l'état final qu'importe le nombre d'exécutions

Gestionnaire de paquets

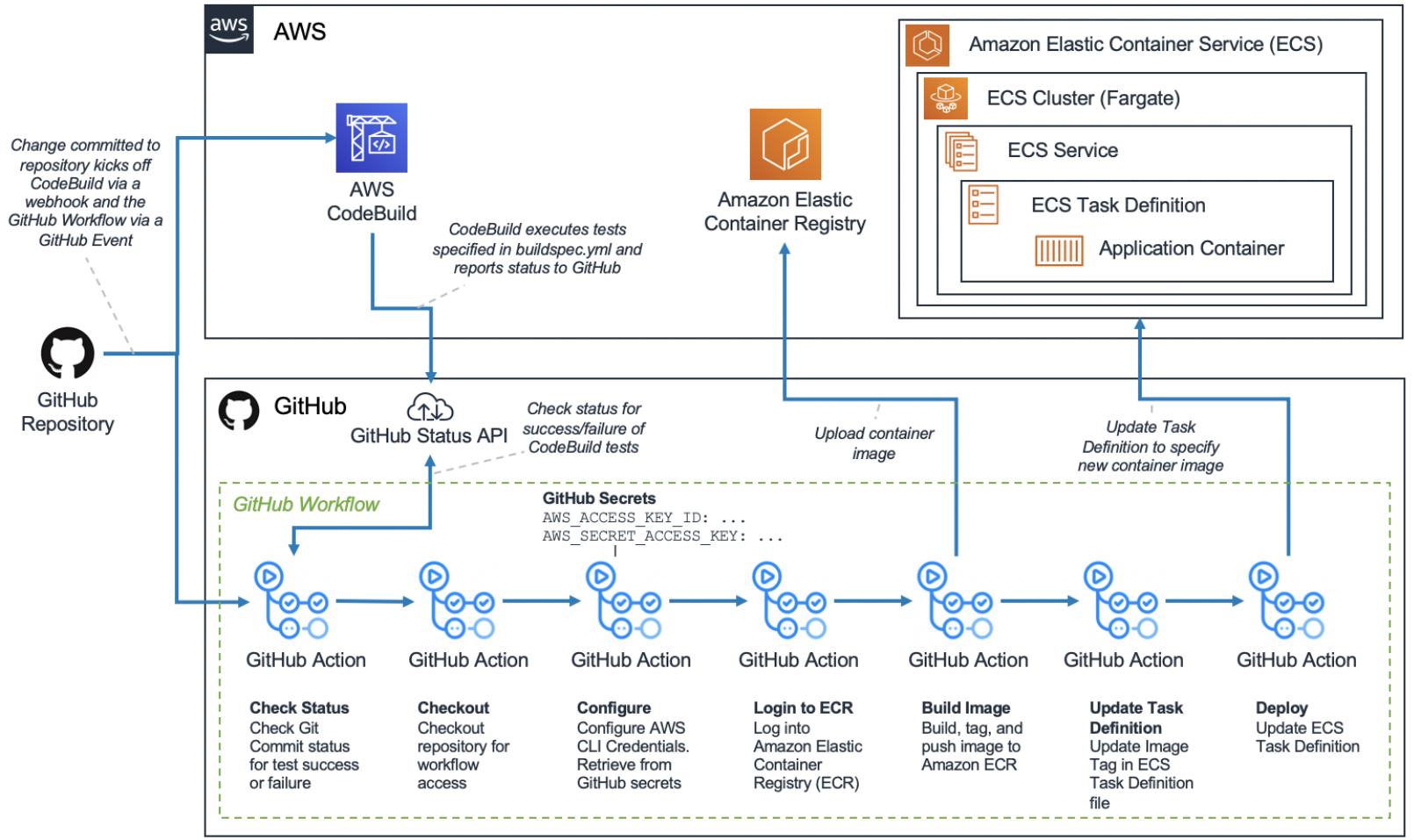
Outil permettant d'automatiser la recherche, l'installation, la désinstallation, la mise à jour et la configuration de paquets

- ✓Facilite la gestion des dépendances d'un projet
- ✓Vérifie la présence de vulnérabilités
- ✓Supprime les paquets fantômes ou inutilisés
- ✓Fais le suivi des dernières mises à jour de tous les paquets

Exemple de pipeline avec Jenkins



CI/CD Pipeline: GitHub Actions + AWS CodeBuild Tests



[Create a CI/CD pipeline for Amazon ECS with GitHub Actions and AWS CodeBuild Tests]