GESTION DE PROJET INFORMATIQUE

À l'origine d'un projet

À l'origine d'un projet

Des besoins

À l'origine d'un projet

Des besoins

Une décision

À l'origine d'un projet

Des besoins

Une décision

De l’argent

À l'origine d'un projet

Des besoins

Une décision

De l’argent

Des utilisateurs

À l'origine d'un projet

Des besoins

Une décision

De l'argent

Des utilisateurs

Des commanditaires

À la réalisation d'un projet

À la réalisation d'un projet

Des développeurs

À la réalisation d'un projet

Des concepteurs

Des développeurs

À la réalisation d'un projet

Des concepteurs

Des développeurs

Des architectes

Des designers

Des administrateurs réseau

À la réalisation d'un projet

Des concepteurs

Des développeurs

Des architectes

Des designers

Des administrateurs réseau

À la réalisation d'un projet

Des concepteurs

Des développeurs

Des architectes

Des chefs d’équipe

Des designers

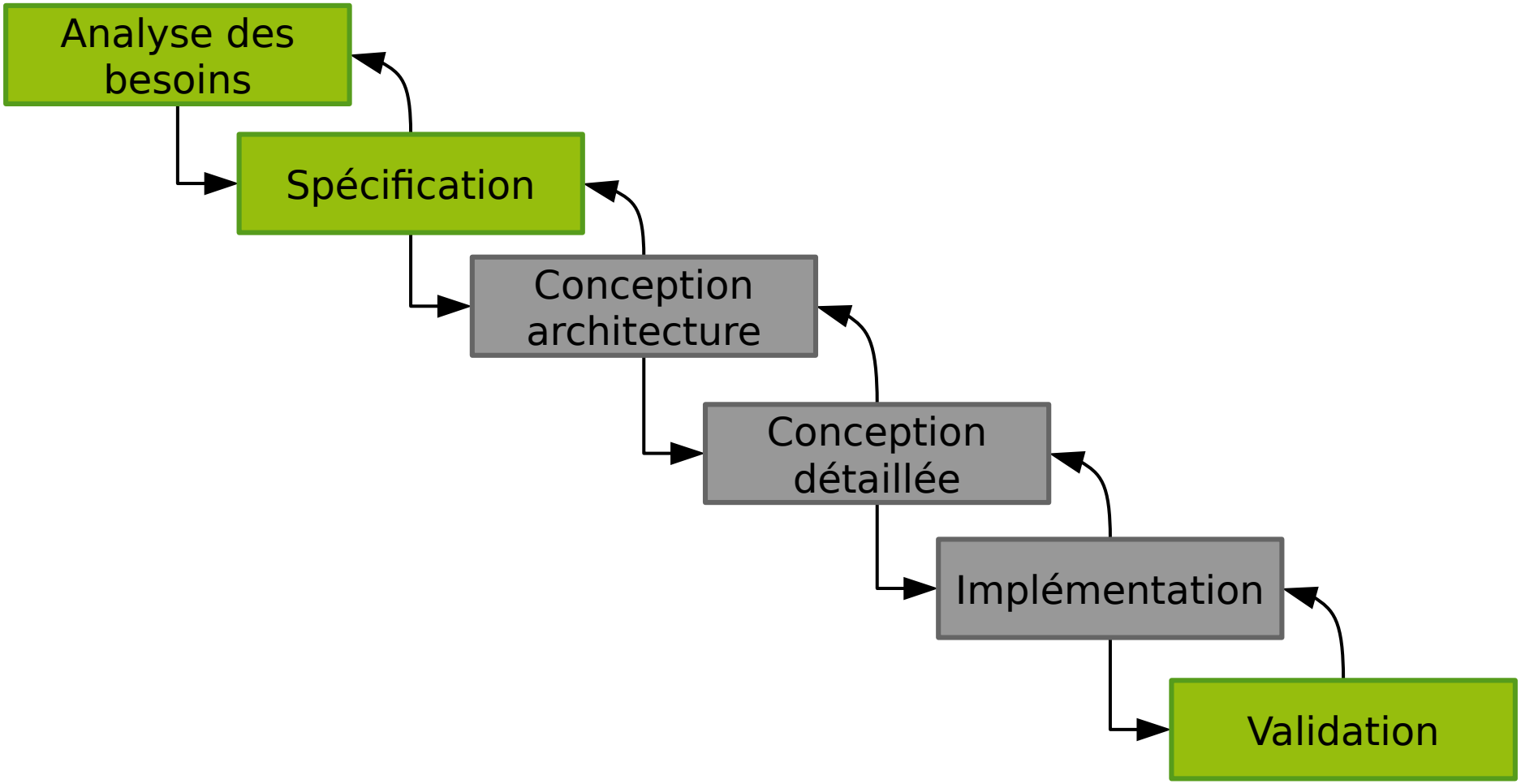
Des administrateurs réseau

Les rôles dans un projet

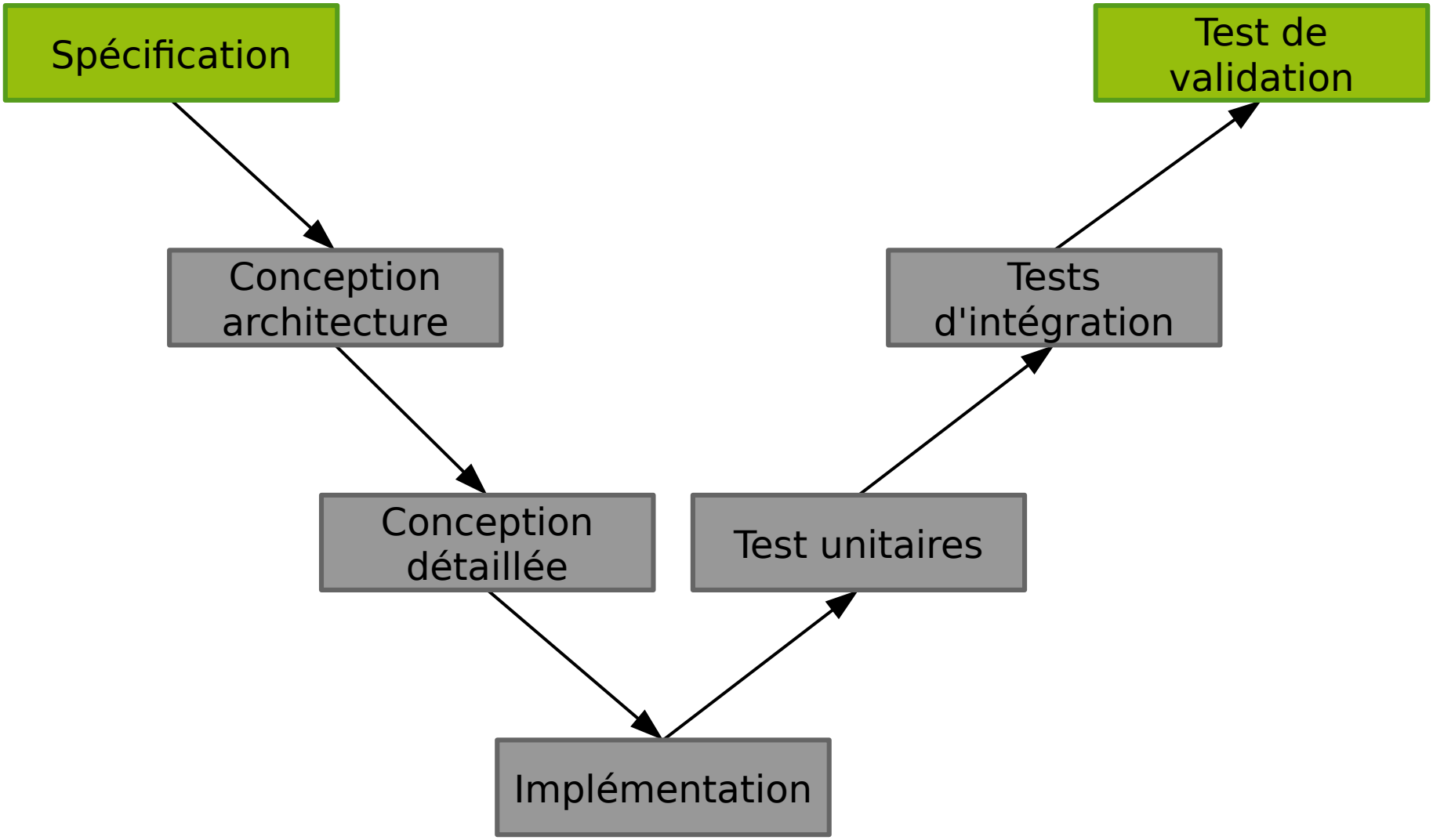
- **La maîtrise d'ouvrage (MOA)** : elle définit le projet, le résultat à obtenir, contrôle la réalisation et valide le résultat final
  - Les utilisateurs : celui pour qui est fait l'ouvrage
  - Le commanditaire : celui qui paie l'ouvrage, il choisit la maîtrise d'œuvre
- **La maîtrise d'œuvre (MOE)** : elle réalise le projet, en respectant les exigences de la MOA (Qualité – Coup - Délai)
  - Les développeurs, analystes, architectes...

Méthodes traditionnelles

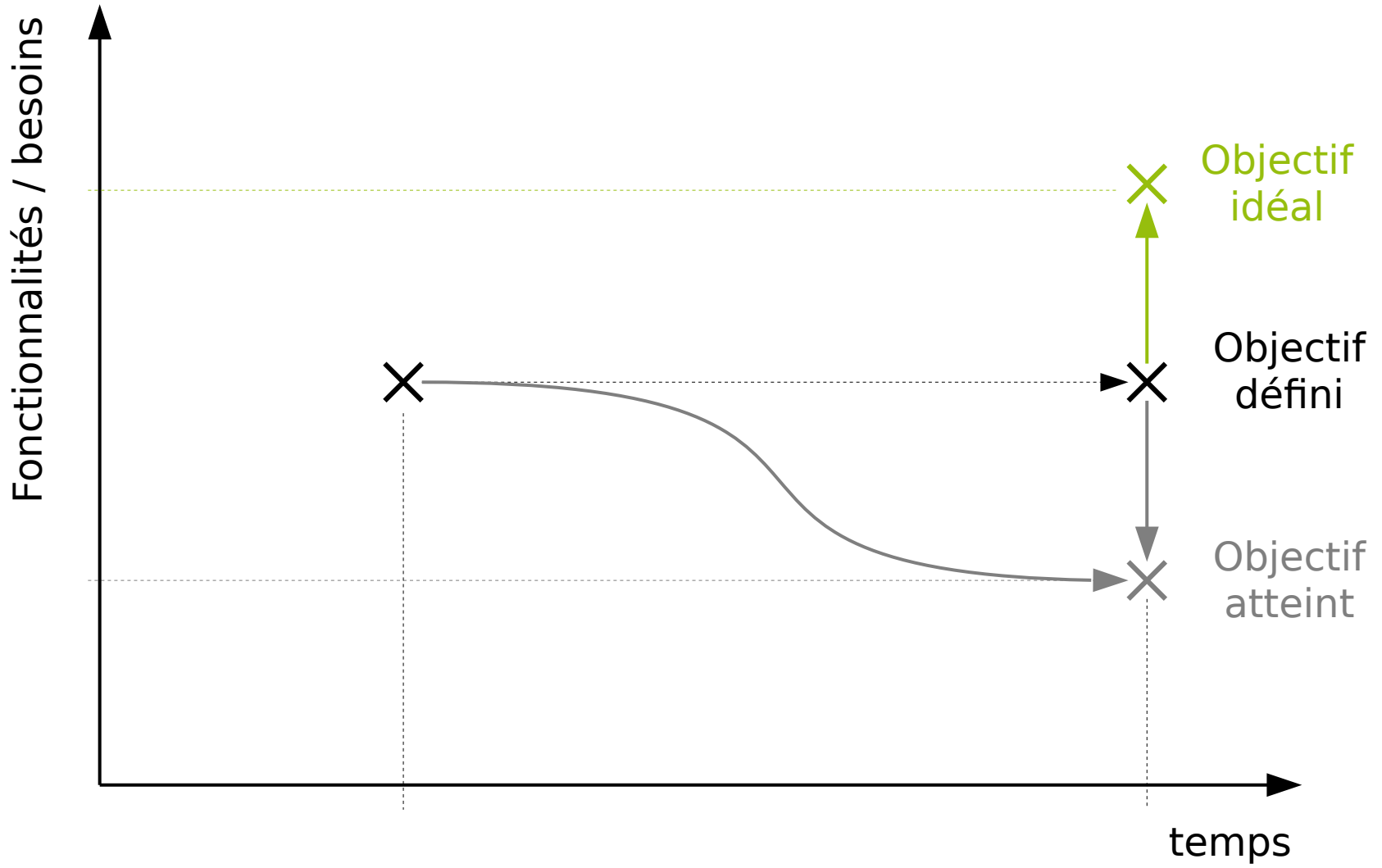
La méthode en cascade



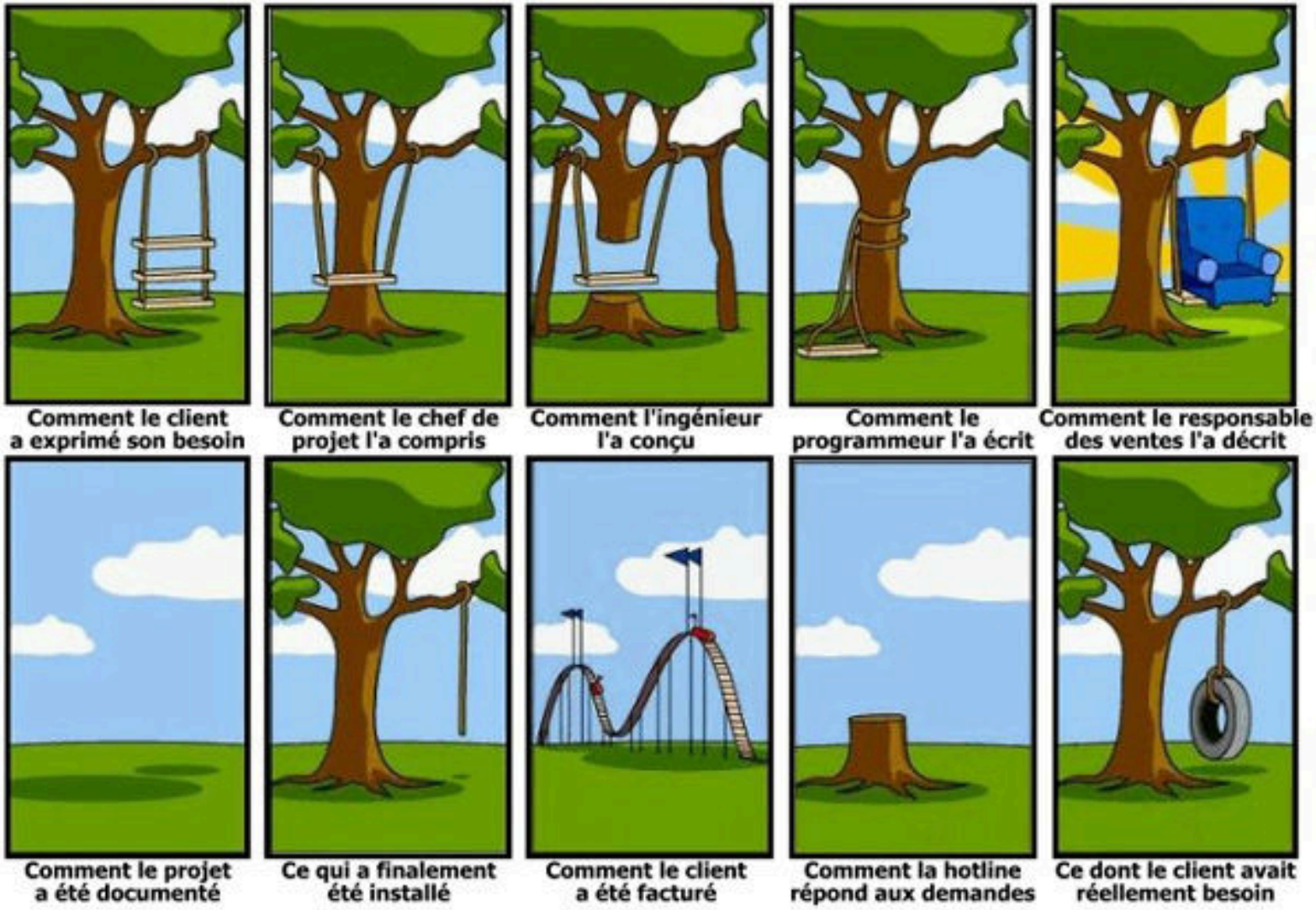
La méthode en V



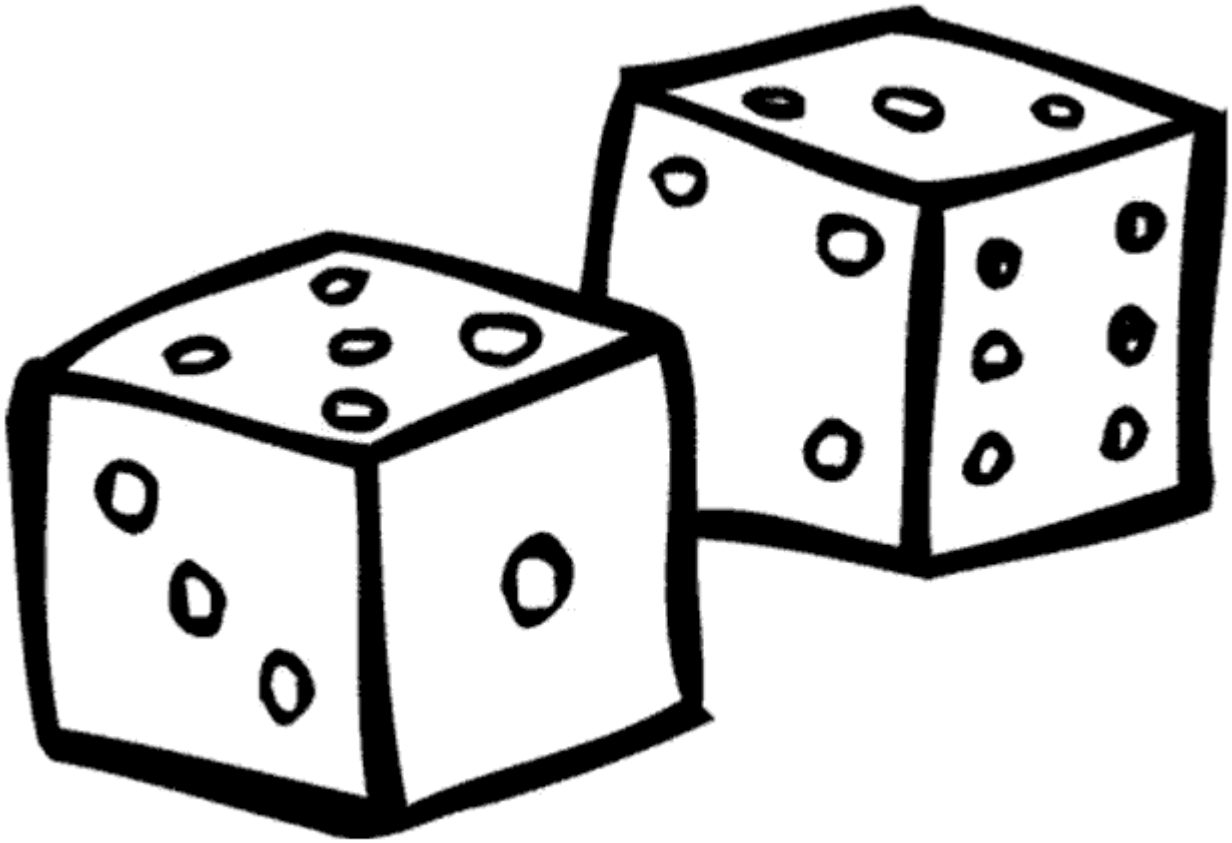
Ce qui arrive bien souvent...



... et pourquoi


Comment le client a exprimé son besoin
Comment le chef de projet l'a compris
Comment l'ingénieur l'a conçu
Comment le programmeur l'a écrit
Comment le responsable des ventes l'a décrit
Comment le projet a été documenté
Ce qui a finalement été installé
Comment le client a été facturé
Comment la hotline répond aux demandes
Ce dont le client avait réellement besoin

Pause jeu



Philosophie Agile

Des 12 principes…

- Notre plus haute priorité est de **satisfaire le client** en livrant **rapidement** et **régulièrement** des fonctionnalités à grande valeur ajoutée.
- Accueillez positivement les **changements** de besoins, même tard dans le projet. Les processus agiles exploitent le changement pour donner un **avantage compétitif au client**.
- Livrez fréquemment un logiciel opérationnel avec des **cycles** de quelques semaines à quelques mois et une préférence pour les plus courts.
- Les utilisateurs ou leurs représentants et les développeurs doivent **travailler ensemble** quotidiennement tout au long du projet.
- Réalisez les projets avec **des personnes motivées**. Fournissez-leur l'environnement et le soutien dont elles ont besoin et faites-leur **confiance** pour atteindre les objectifs fixés.
- La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le **dialogue** en face à face.
- Un **logiciel opérationnel** est la principale mesure d'avancement.
- Les processus agiles encouragent un rythme de développement soutenable. **Ensemble**, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.
- Une attention continue à l'**excellence technique** et à une bonne conception renforce l'agilité.
- La **simplicité** – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.
- Les meilleures architectures, spécifications et conceptions émergent d'équipes **auto-organisées**.
- À intervalles réguliers, l'**équipe réfléchit** aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.

… aux 4 valeurs

- **Les individus et leurs interactions**

  plus que les processus et les outils.

- **Du logiciel qui fonctionne**

  plus qu’une documentation exhaustive.

- **La collaboration avec les clients**

  plus que la négociation contractuelle.

- **L’adaptation au changement**

  plus que le suivi d’un plan.

Un peu de vocabulaire

- **Cycle / Itération / Sprint :** période de 1 à 4 semaines pendant lesquelles la MOE réalise un sous ensemble des fonctionnalités.

  → un livrable fonctionnel, testable

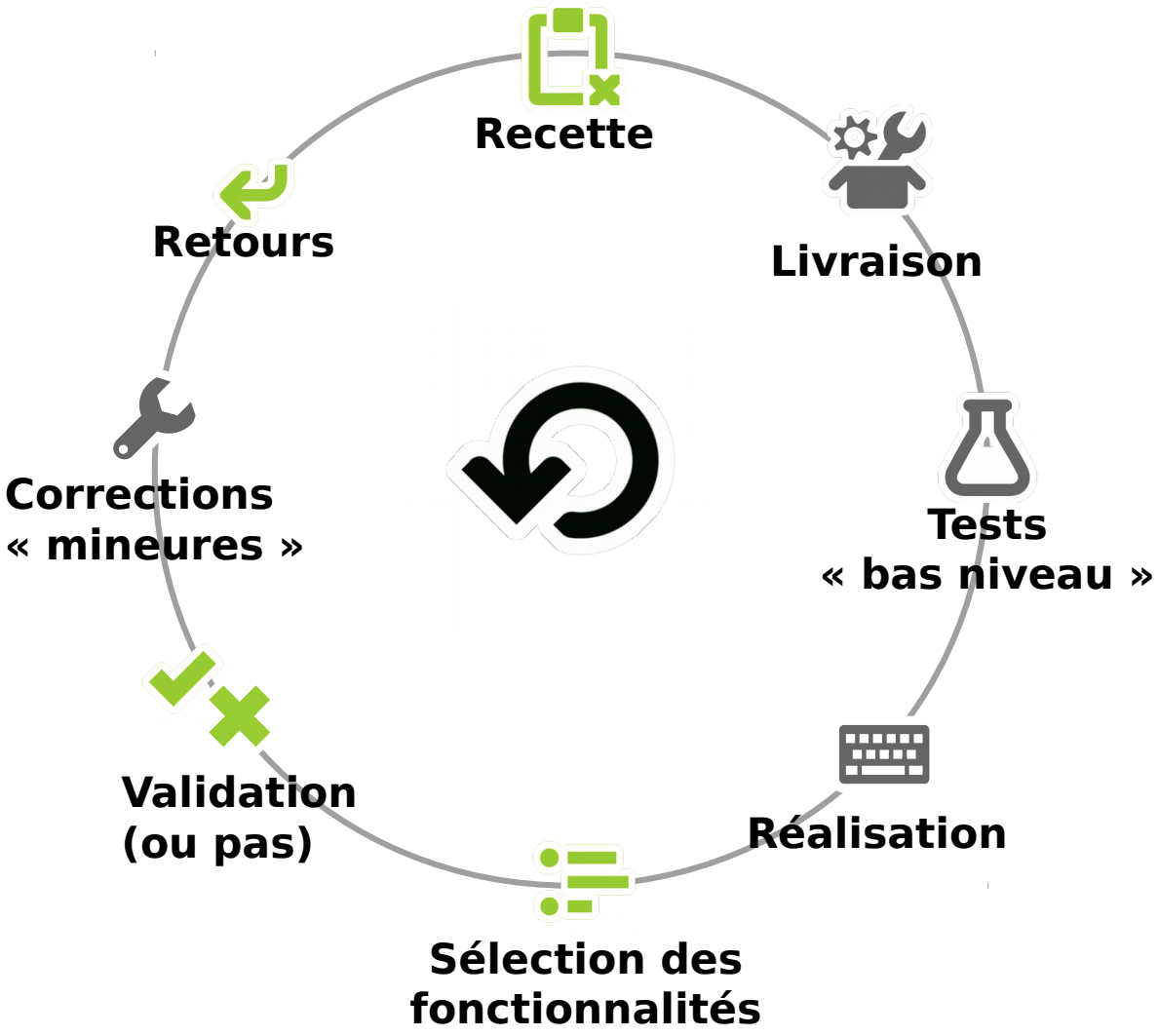
- **Fonctionnalité / Scénario / User story :** un composant du logiciel, répondant à un besoin de l'utilisateur final

- **Tâche :** un développement élémentaire

- **Product backlog :** ensemble des fonctionnalités restantes

- **Feedback :** retour des avis des utilisateurs sur le livrable logiciel actuel

Un cycle


Recette
Livraison
Tests
« bas niveau »
Réalisation
Sélection des
fonctionnalités
Validation
(ou pas)
Corrections
« mineures »
Retours

À chacun sa vision

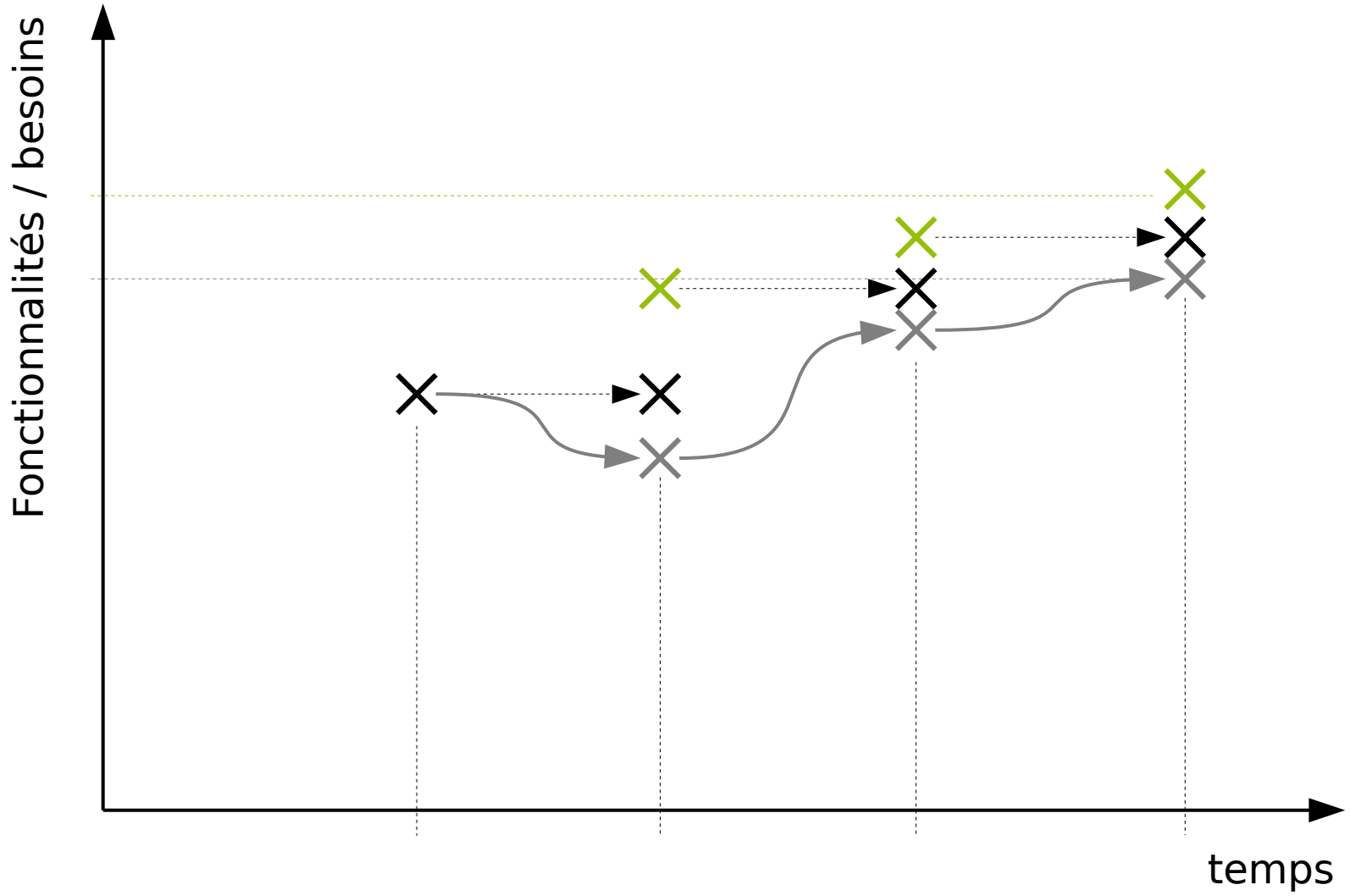
Estime
(business value)

1 besoin / fonctionnalité

Estime
(temps de développement)

N tâches

Ce que ça peut donner



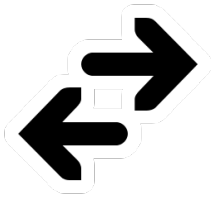
Méthode Agile : Scrum

Un peu de vocabulaire

- **Sprint**
- **Product owner :** représentant des utilisateurs, de la MOA, en contact permanent avec l'équipe de réalisation
- **Scrum master :** membre de l'équipe de développement, assurant l'application de la méthodologie Scrum. Il est l'interlocuteur privilégié du product owner. Il coordonne si besoin les différentes équipes.
- **Équipe Scrum :** les développeurs, jusqu'à 10 personnes
- **Mélée quotidienne :** 5 – 10 minutes tous les jours, pour faire un point

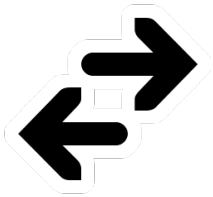
Scrum : les acteurs et les interactions

Utilisateurs



Product Owner

Scrum master



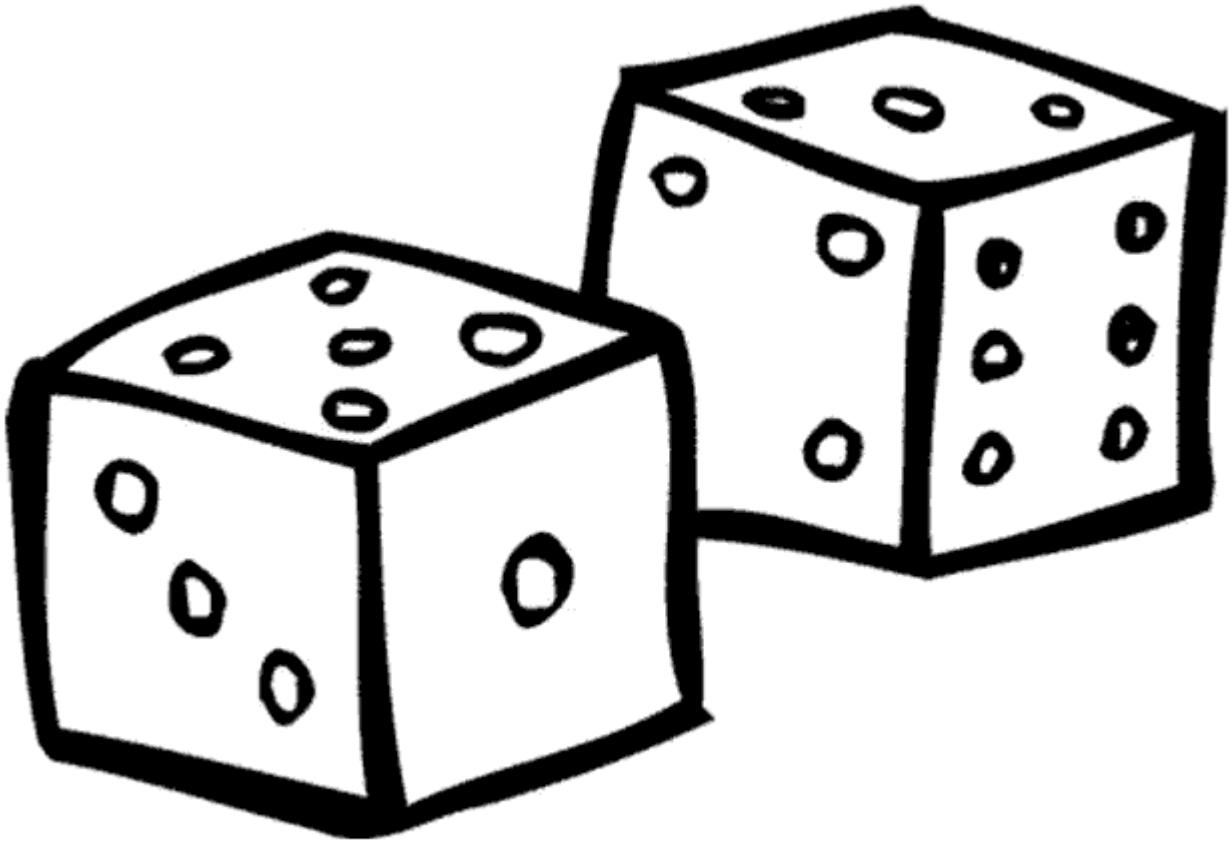
Développeurs

Méthode Agile : Extrem Programming

Extrem Programming : les pratiques

- On pousse les bonnes pratiques de développement à l'extrême
  - **Revue de code →** On développe à deux (Pair programming)
  - **Tests →** on commence par ça (Test Driven Development)
  - **Simplicité du code →** la ligne de conduite (Keep It Stupid Simple)
  - **Code bien conçu →** refactoring permanent
- Des cycles très rapides

Pause jeu



Comparons

Comparons

Agile

- Engagement sur les moyens
  - Au forfait par itération
  - Échanges de fonctionnalités
- **Risque** : pas de documentation précise
- Une application plus modulaire, plus évolutive
- **Risque** : un développement qui n'avance pas

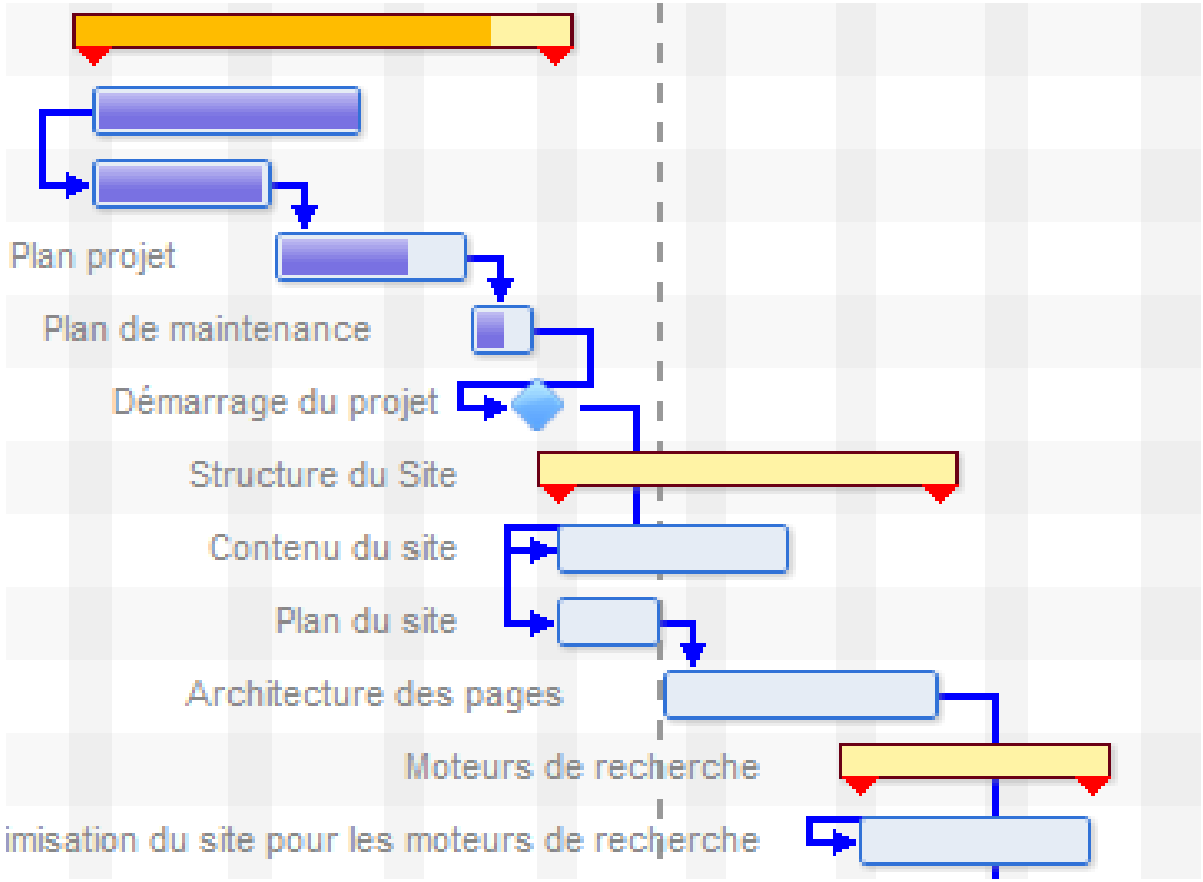
Classique

- Engagement sur le résultat (au forfait)
  - Avenant
  - Tranches conditionnelles
- Documentation obligatoire, qui peut servir de référence
- Une solution finale statique (potentiellement monolithique)

Outils et pratiques

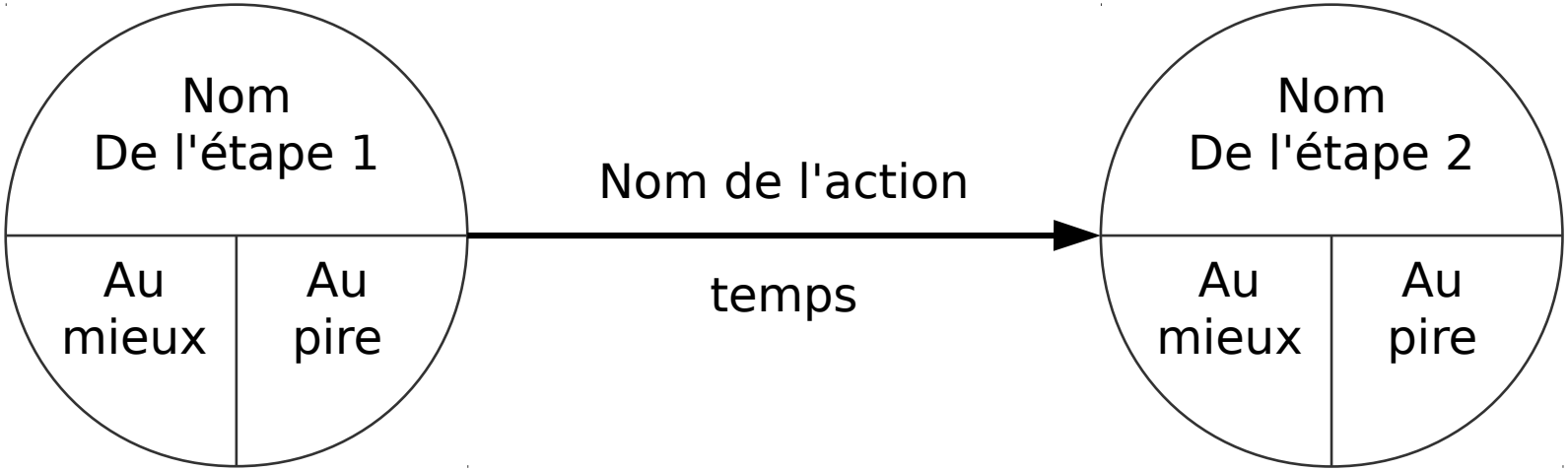
Le GANTT

- Pour planifier les tâches

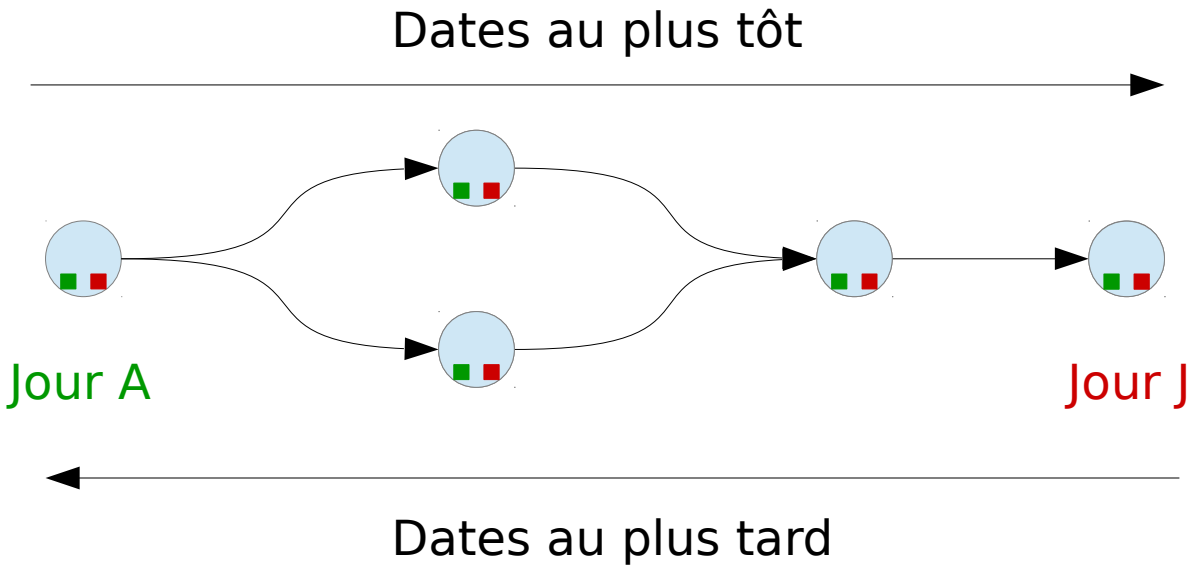


Le PERT

- Graphe des tâches
  - Pour lister les tâches
  - Pour ordonner les tâches (dépendances)
  - Pour estimer les dates
  - Pour connaître le chemin critique

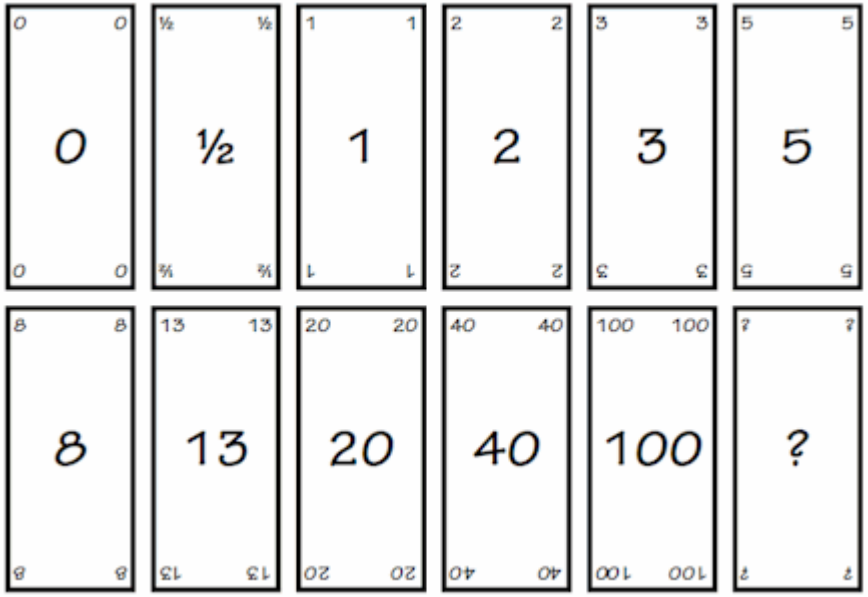


Le PERT



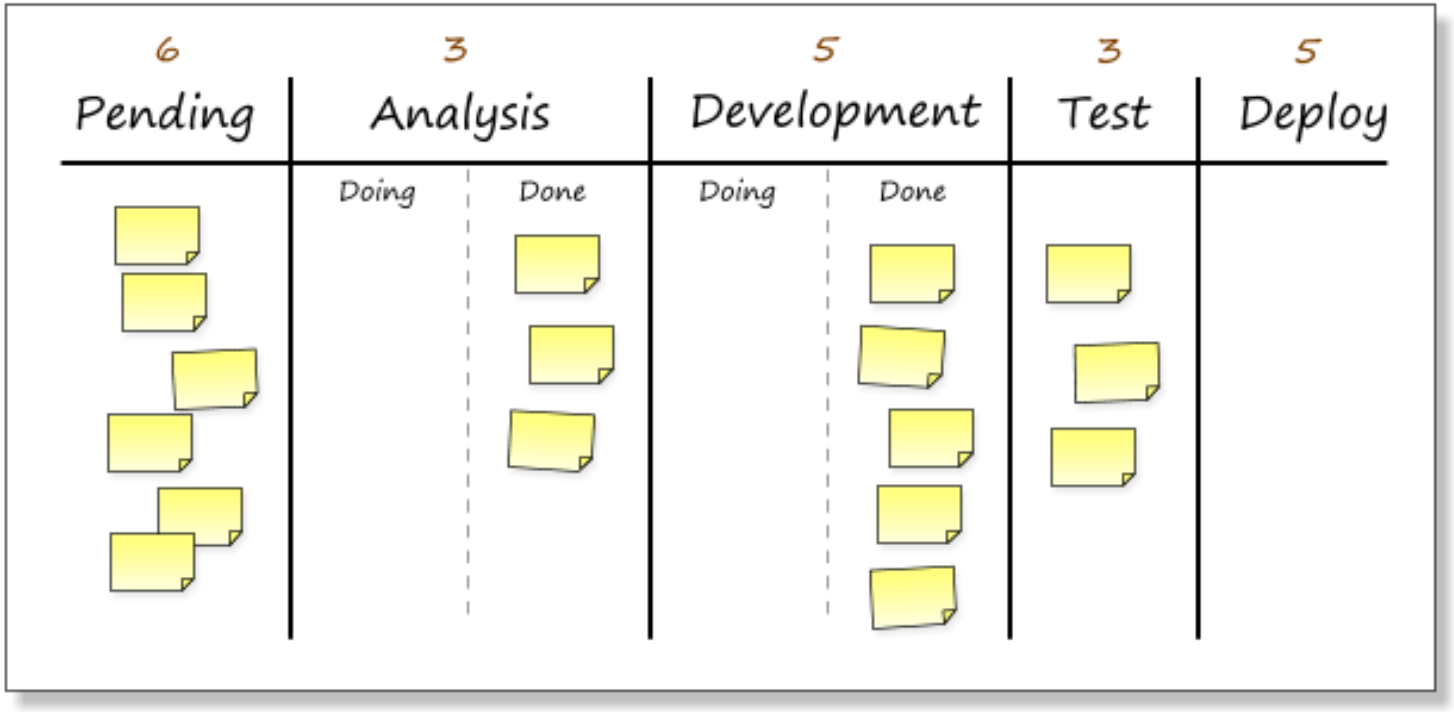
Le poker planning

- Pour estimer collégialement le temps de développement d'une tâche
  - Avoir des cartes



Le Kanban

- Pour visualiser où on en est



Un Pert presque parfait !

- Préparer un repas : Pert et Poker planning

Des logiciels

- Pour communiquer : un gestionnaire de tickets (bug tracker)

- Pour partager le code : un gestionnaire de version de code

- Pour automatiser les livraisons : un outil d'intégration continue

Retour aux sources

Du beau code

- Du code qui tourne
  - Tests unitaires / fonctionnels

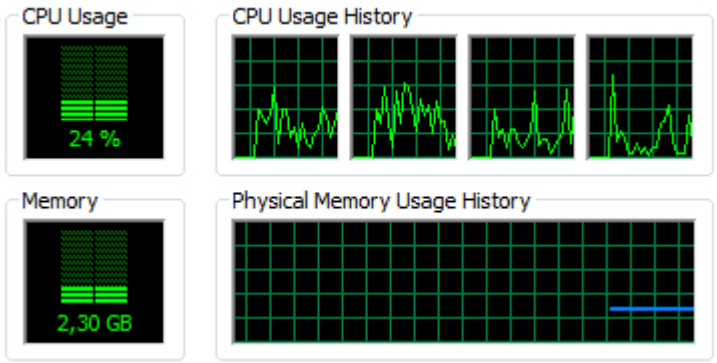
    → on « évite » les régressions

    → ça sert de spécification
  - Tests de déploiement
- Du code propre
  - Pas de code dupliqué
  - Des fonctions pas trop longues
  - Pas de code mort

→ conservation ou amélioration de la qualité du code

Du beau code

- Alors, on perd la mémoire
  - Peut avoir des effets catastrophique
  - Dépend des langages (garbage collector)
  - Outils d'analyse de la mémoire : valgrind
  - Dans tous les cas, on fait gaffe à la mémoire



Du beau code

- RTFM je veux bien mais alors on documente
  - Documentation développeur
    - Commentaires, graphes d'appels, diagrammes
    - Documentation automatique : doxygen, naturaldocs
  - Documentation utilisateur
    - Tutoriels
    - Posters

Passons à la pratique

Pour chaque projet

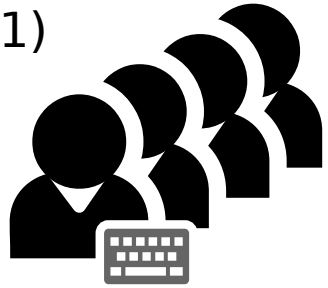
- 4 sujets possibles
  - Un CV (HTML + CSS)
  - Un site carto avec Openlayers 3 (HTML + CSS + javascript)
  - Un site carto avec Leaflet (HTML + CSS + javascript)
  - Un site visu 3D avec iTowns (HTML + CSS + javascript)
- ~10 étudiants

Utilisateurs (2)

Product Owner (1)

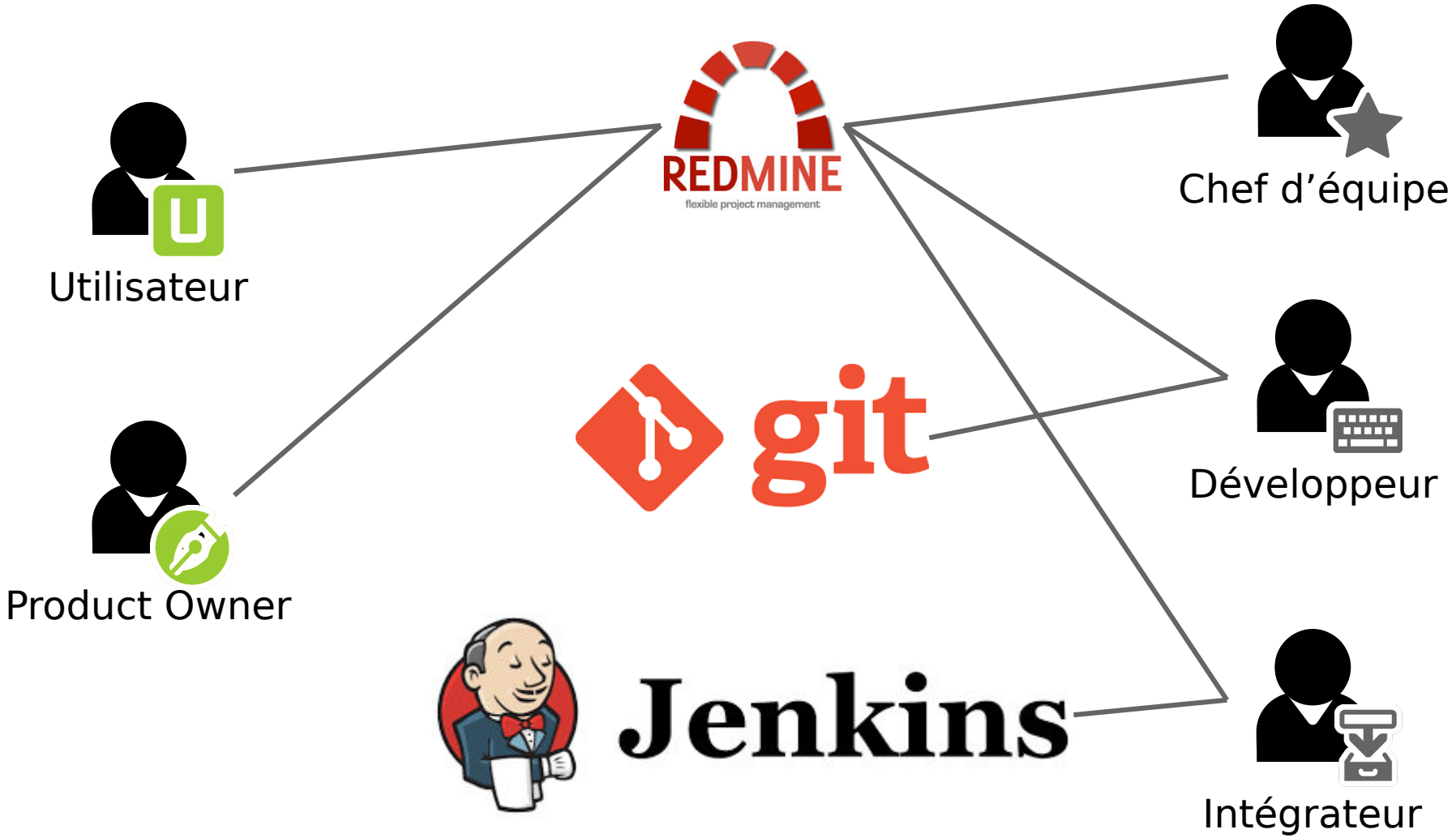
Intégrateurs (2)

Chef d'équipe (1)

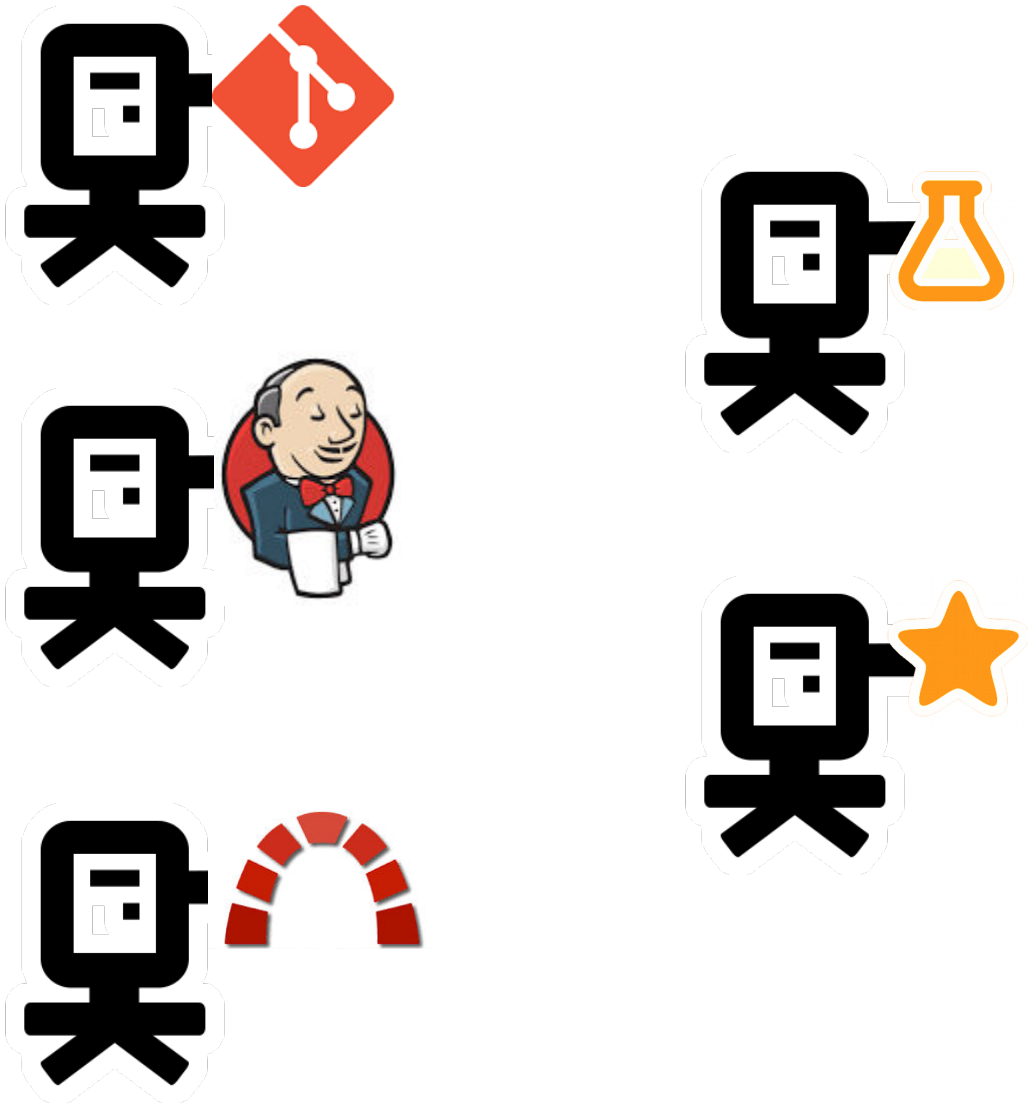


Développeurs (2x2)

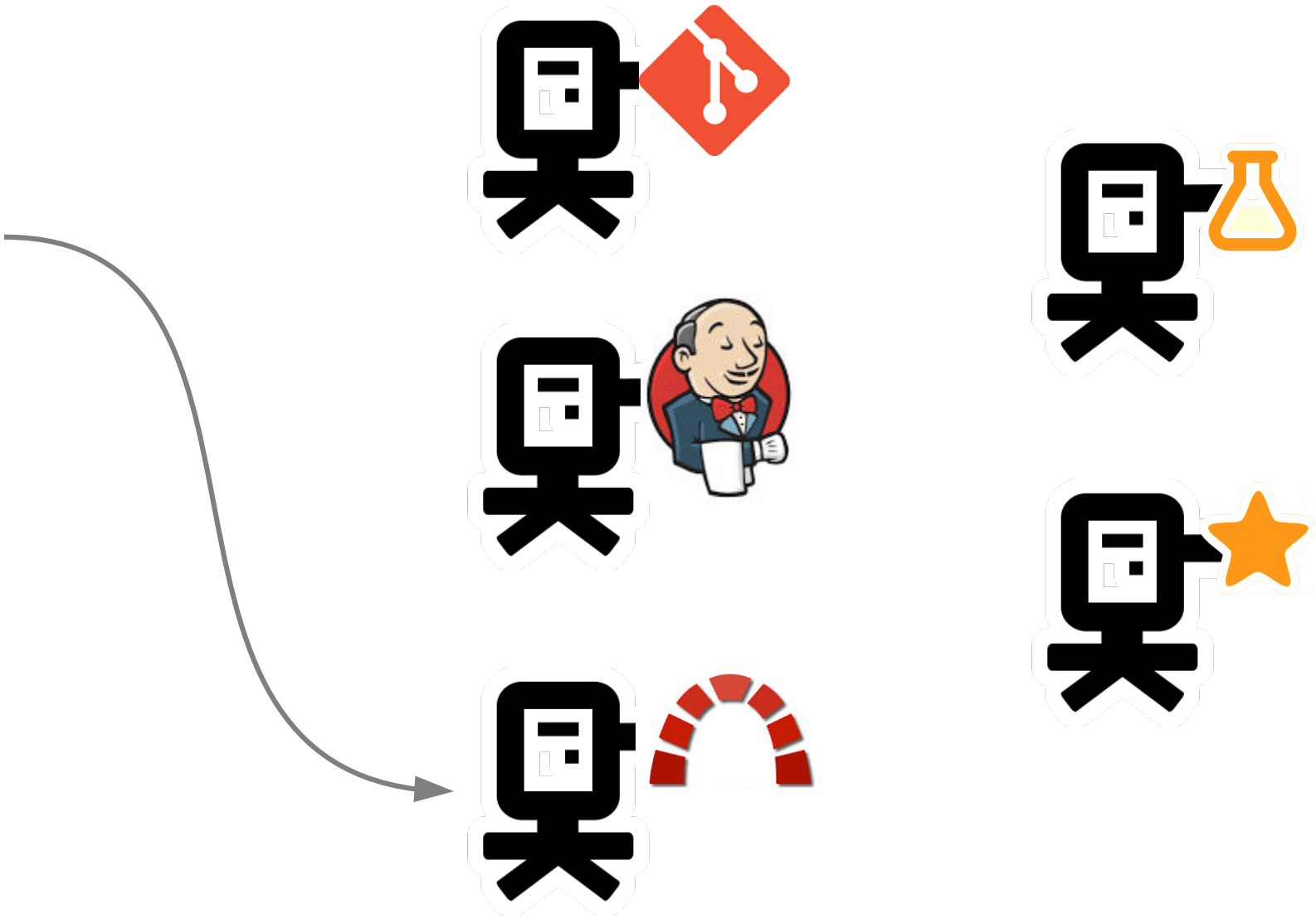
Outils manipulés



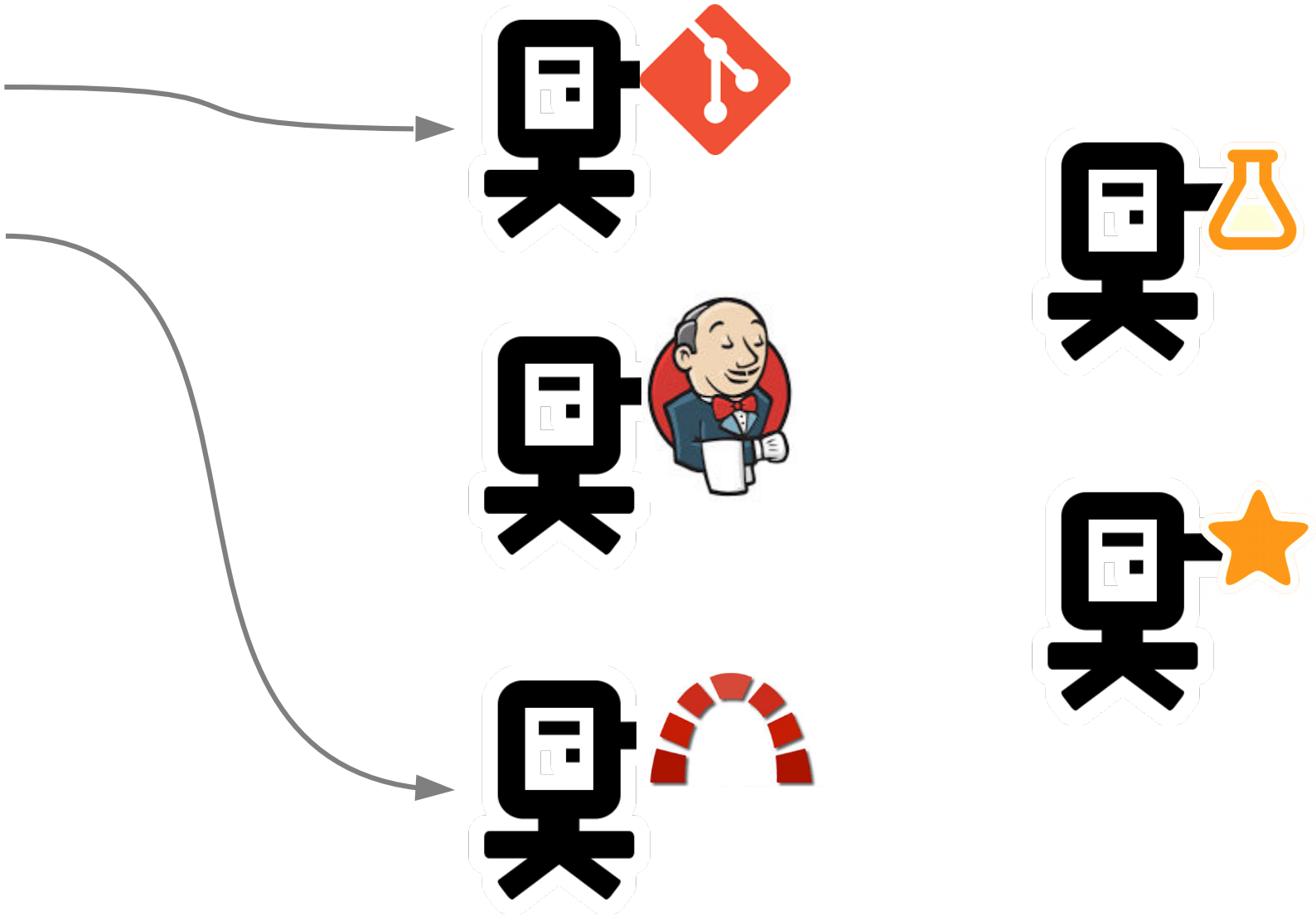
Architecture et déroulement



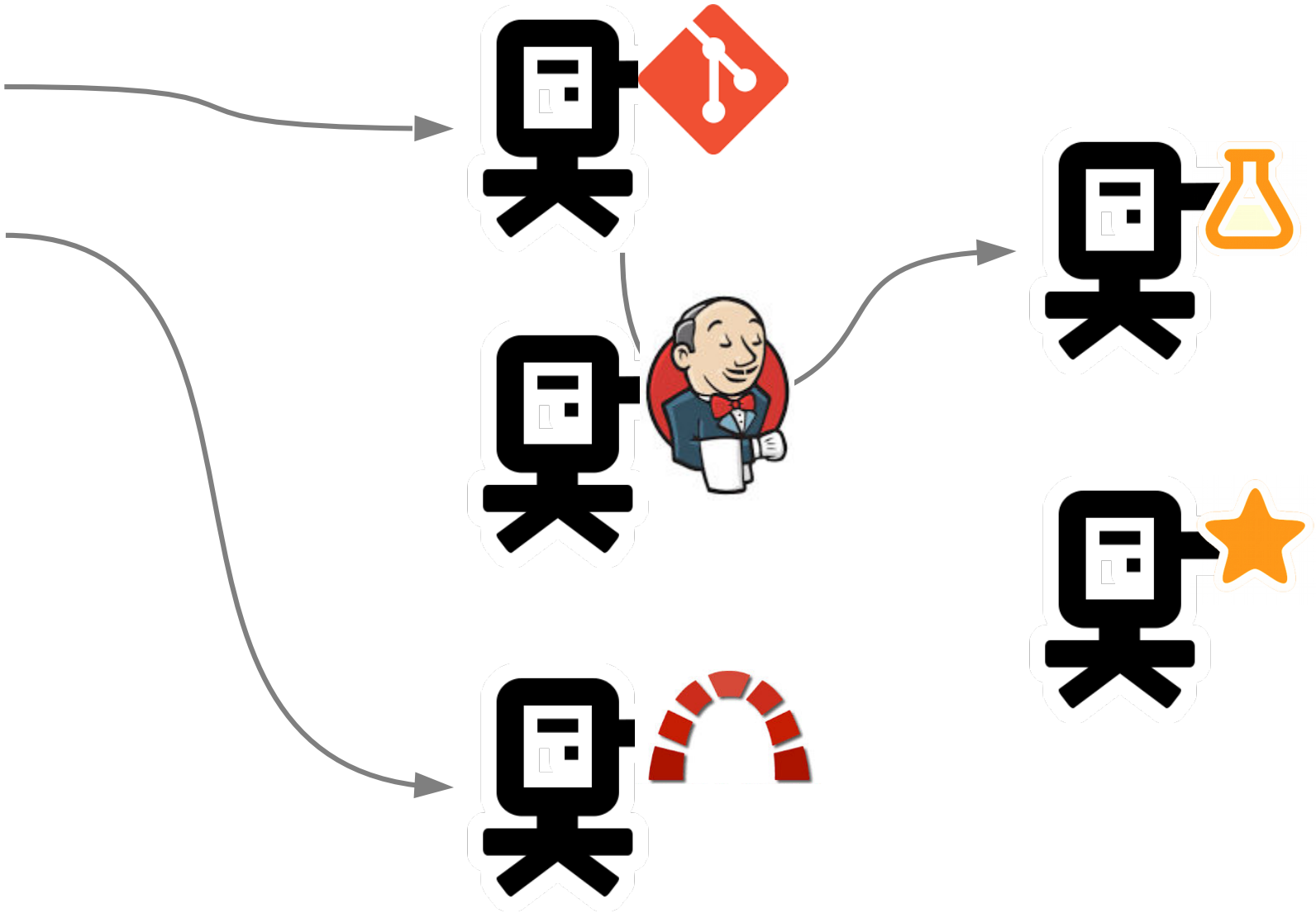
Architecture et déroulement



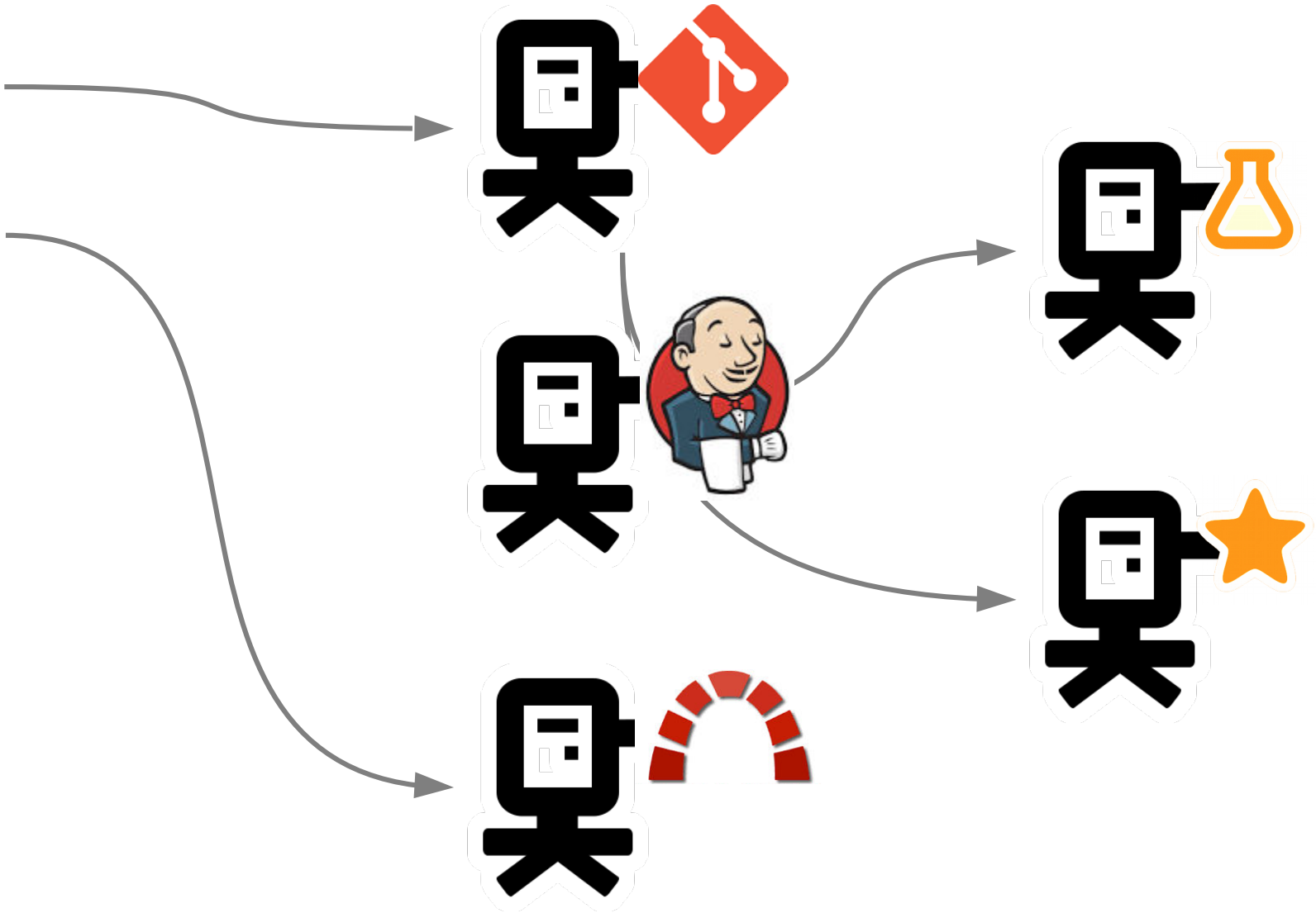
Architecture et déroulement



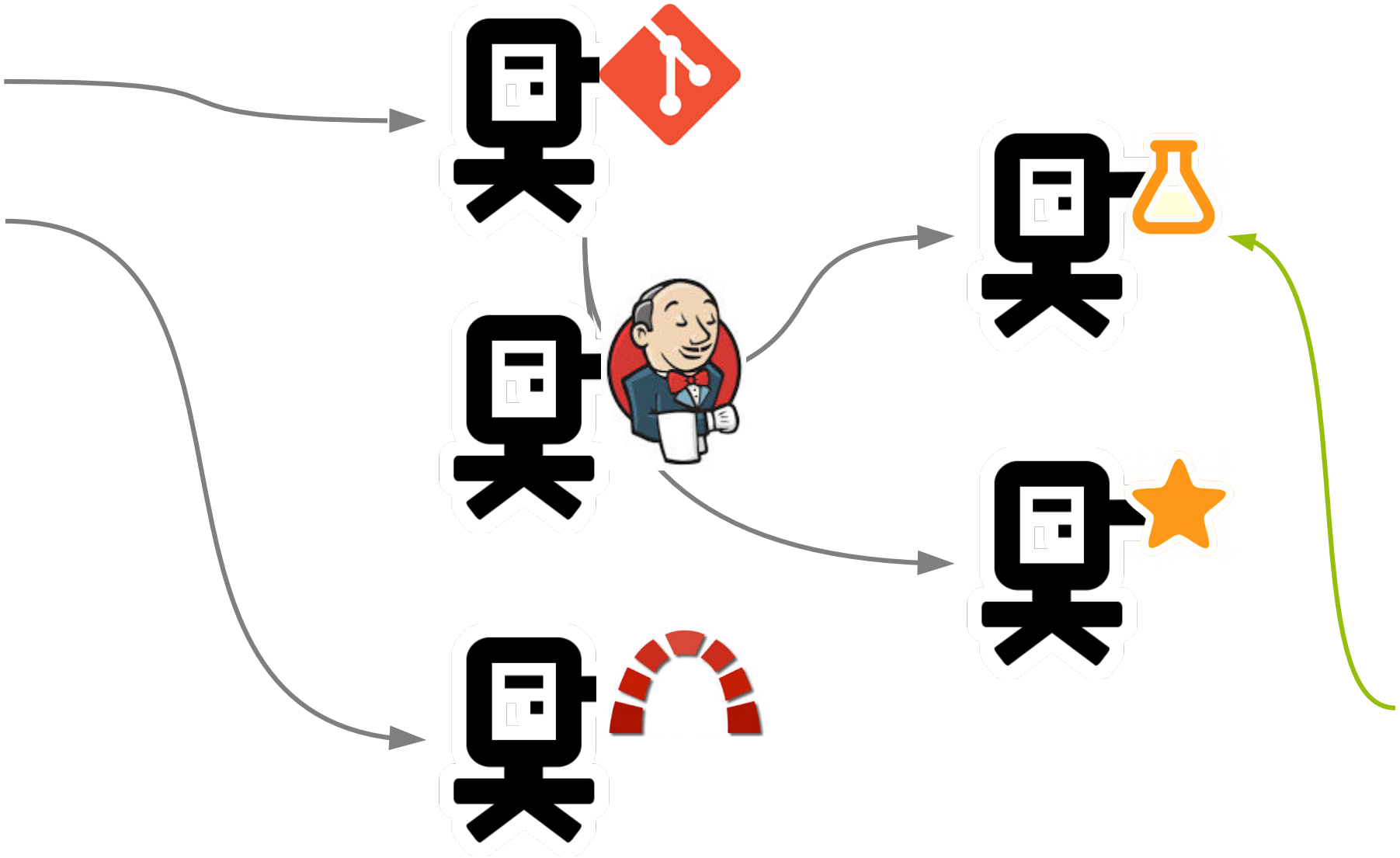
Architecture et déroulement



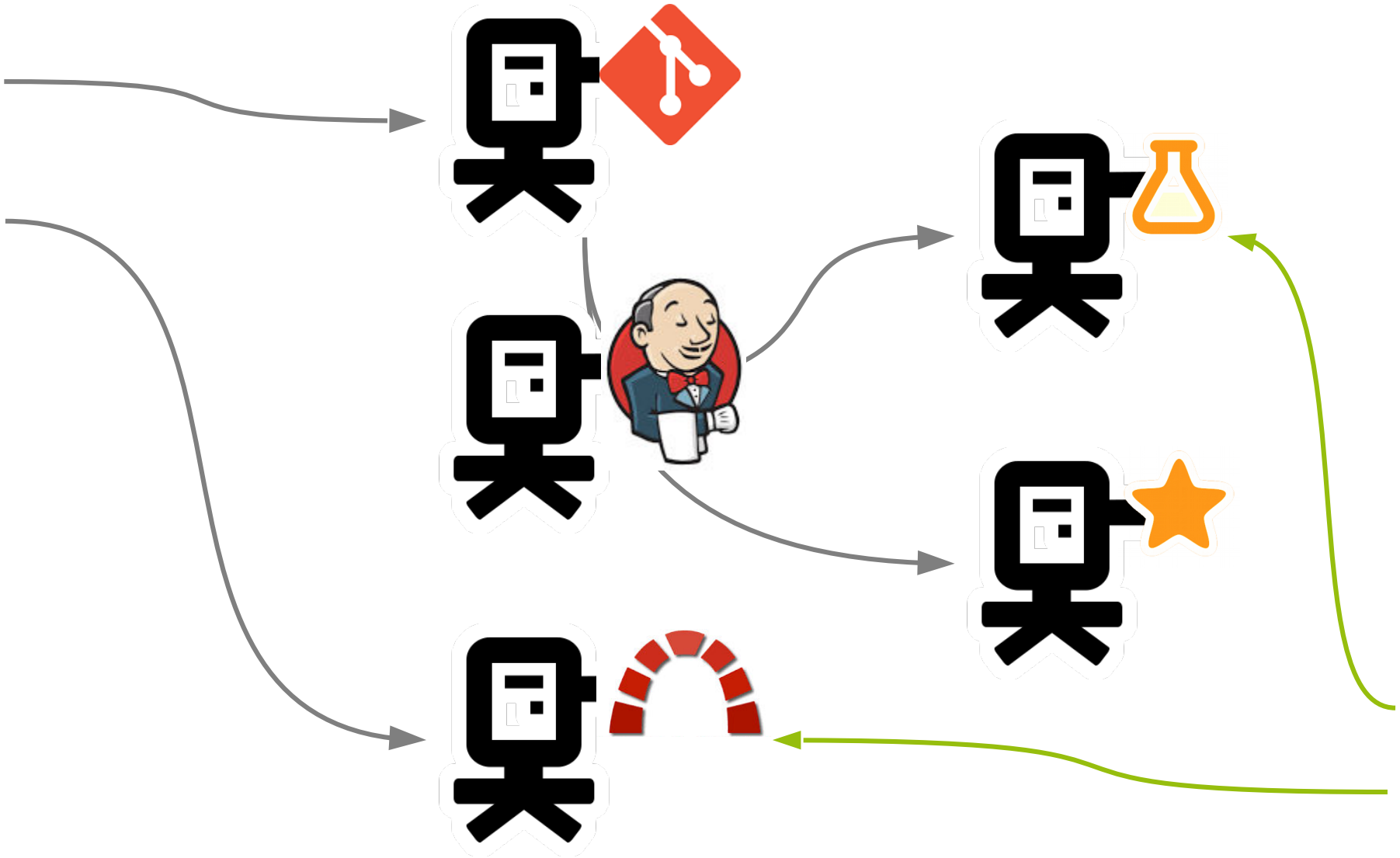
Architecture et déroulement



Architecture et déroulement



Architecture et déroulement



À vous de jouer