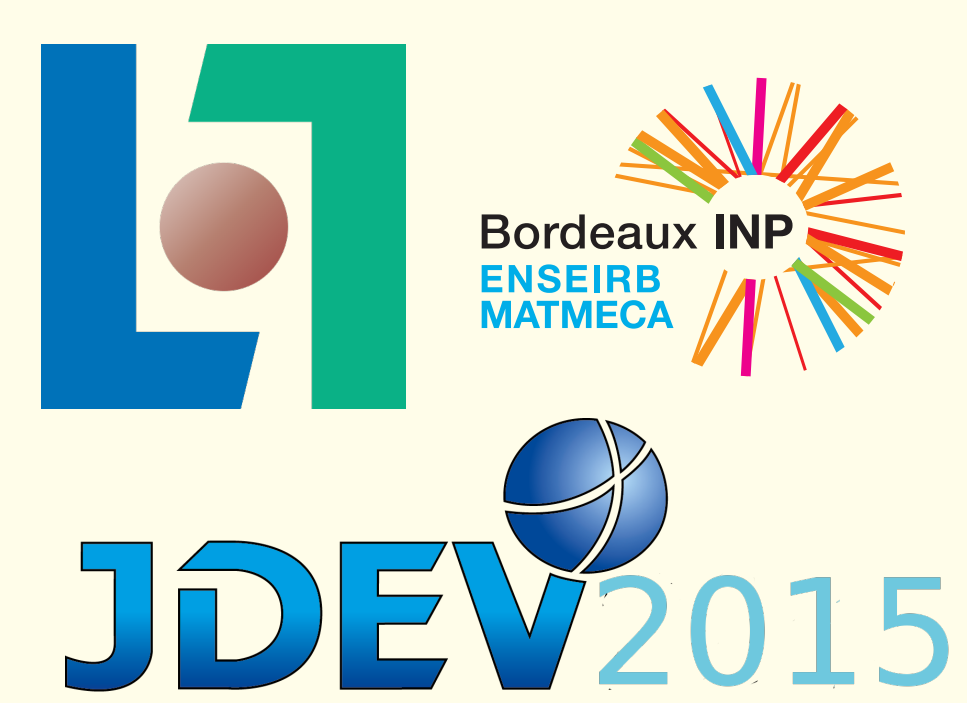


Développer, Tester et Livrer des Microservices à l'aide de Docker

Nicolas Herbaut (LaBRI, ENSEIRB-MATMECA)
David Bourasseau et Maxime Peterlin (ENSEIRB-MATMECA)



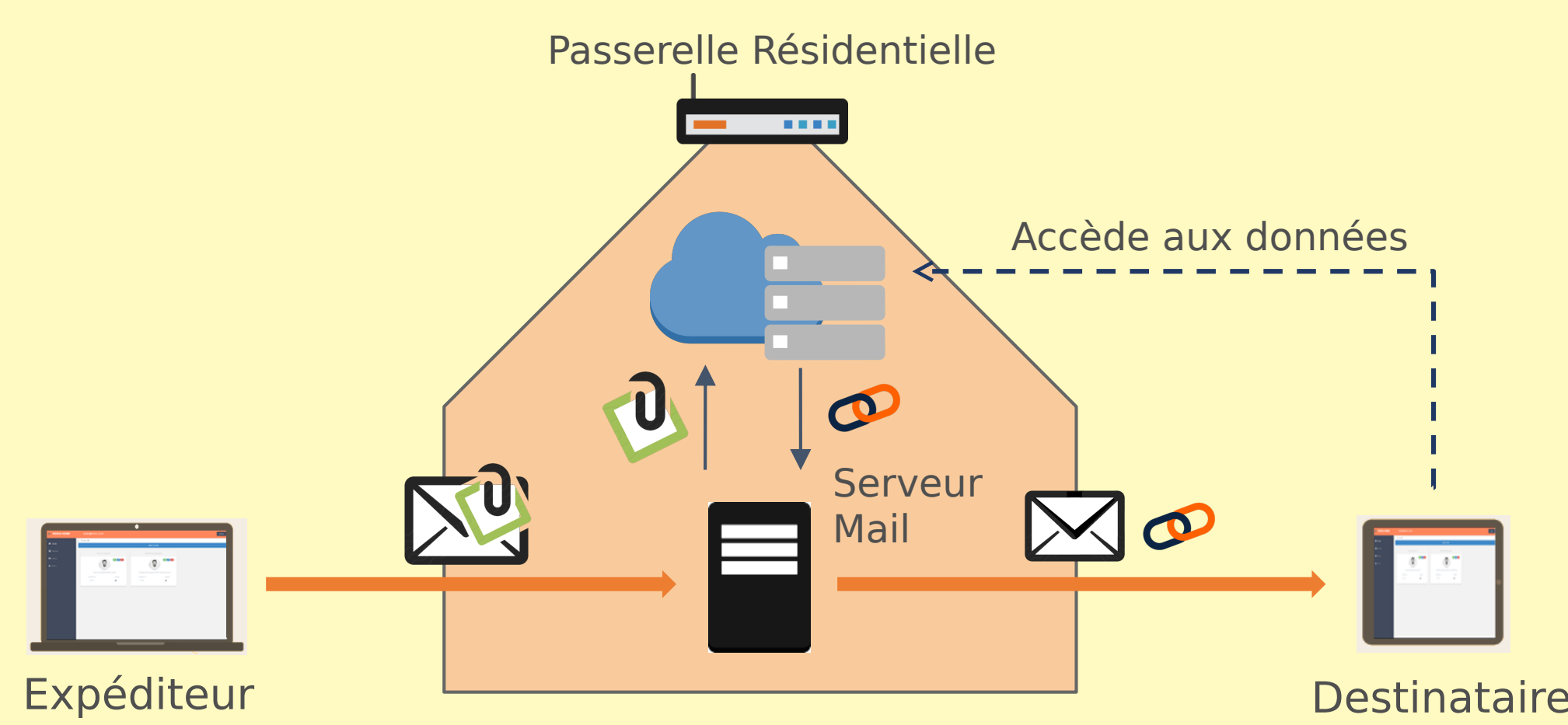
Contribution

Nous exposons un **retour d'expérience** sur le déroulé de 2 projets étudiants où nous avons utilisé Docker au service d'une architecture Microservice.

Nous expliquons comment nous avons utilisé Docker comme **unité de déploiement** pour les mises en production et comme **unité de développement** pour les tests et l'Intégration Continue.

Mots clés: Docker, Architecture Microservice, Architecture RestFul, Intégration Continue, Jenkins, Tests Unitaires, Tests d'intégration, Java, Mock.

Cobaye : Le Projet SnapMail

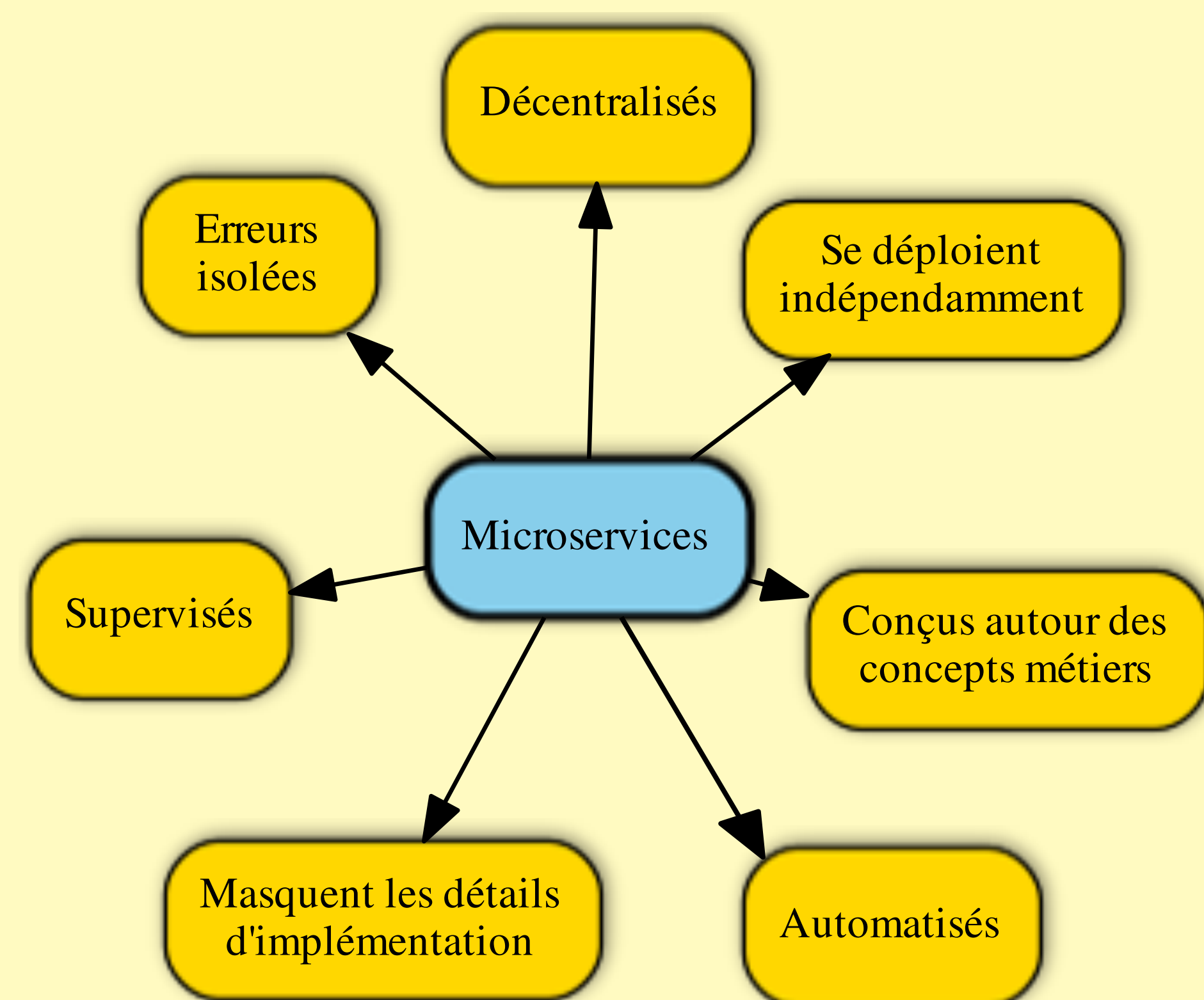


Plateforme privée d'hébergement de pièces jointes
Pour pallier au problème des **pièces jointes** vo-

lumineuses, un serveur SMTP est déployé sur une passerelle résidentielle. L'utilisateur configure son client mail avec ce serveur afin d'y faire transiter son courrier sortant. La passerelle va **détacher les pièces jointes, les stocker et les remplacer par des liens** dans le mail. Ces liens pointent vers les contenus originaux ainsi que vers des **versions optimisées** des vidéos (streaming adaptatif) et photos (converties et redimensionnées).

Le projet est conçu comme un greffon à un autre projet de **réseau social distribué** également déployé sur les passerelles résidentielles.

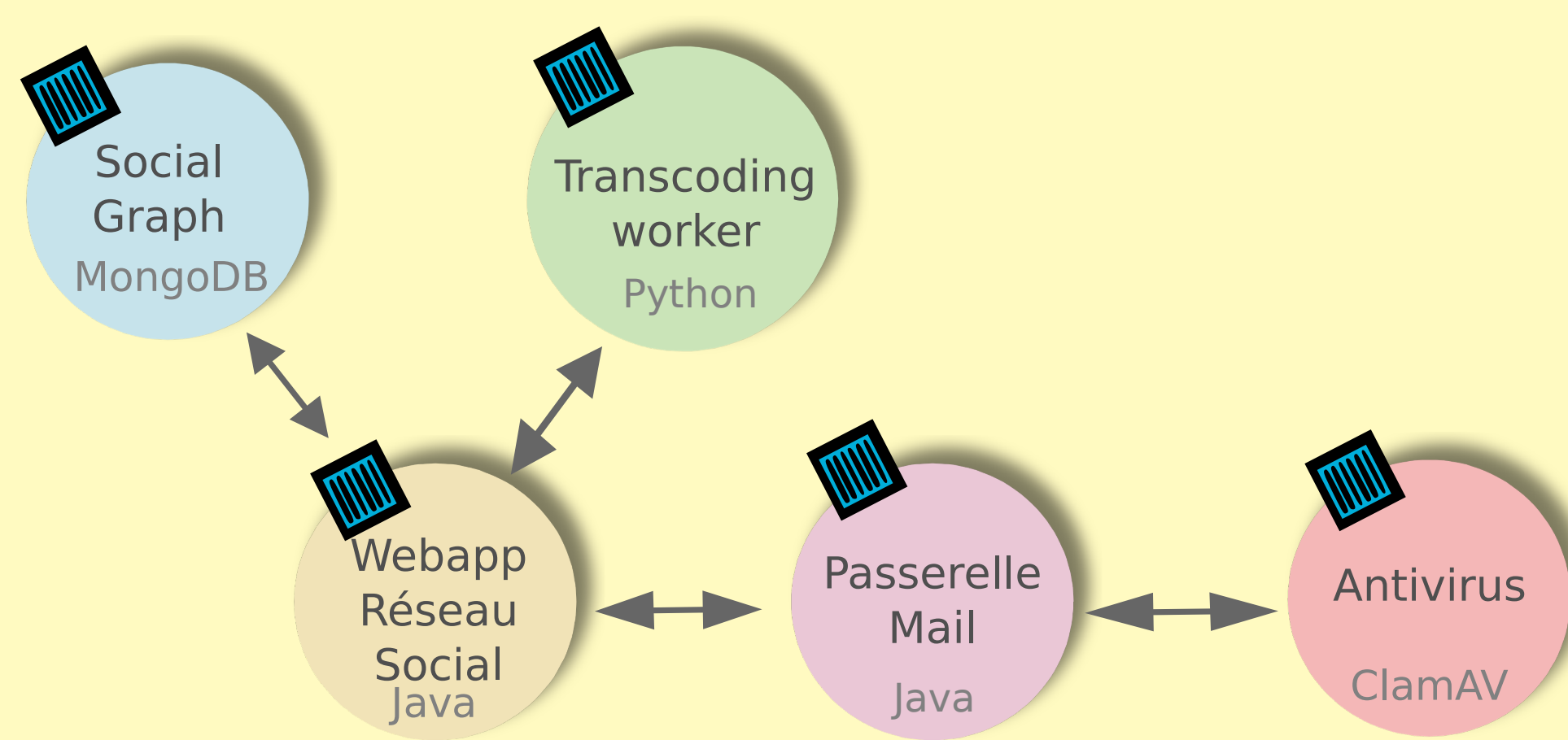
Bénéfices des Microservices



Principes guidant les architectures Microservices.

Les MS sont des **Services petits et autonomes qui fonctionnent ensemble**, promouvant un couplage lâche et une cohésion forte. Les MS émergent des tendances actuelles de l'ingénierie logicielle (*Design Driven Development, Livraison Continue, Virtualisation, Agilité, mouvement DevOps, Antifragile*) etc.

Mise en place dans nos projets



Découpage selon les concepts métiers

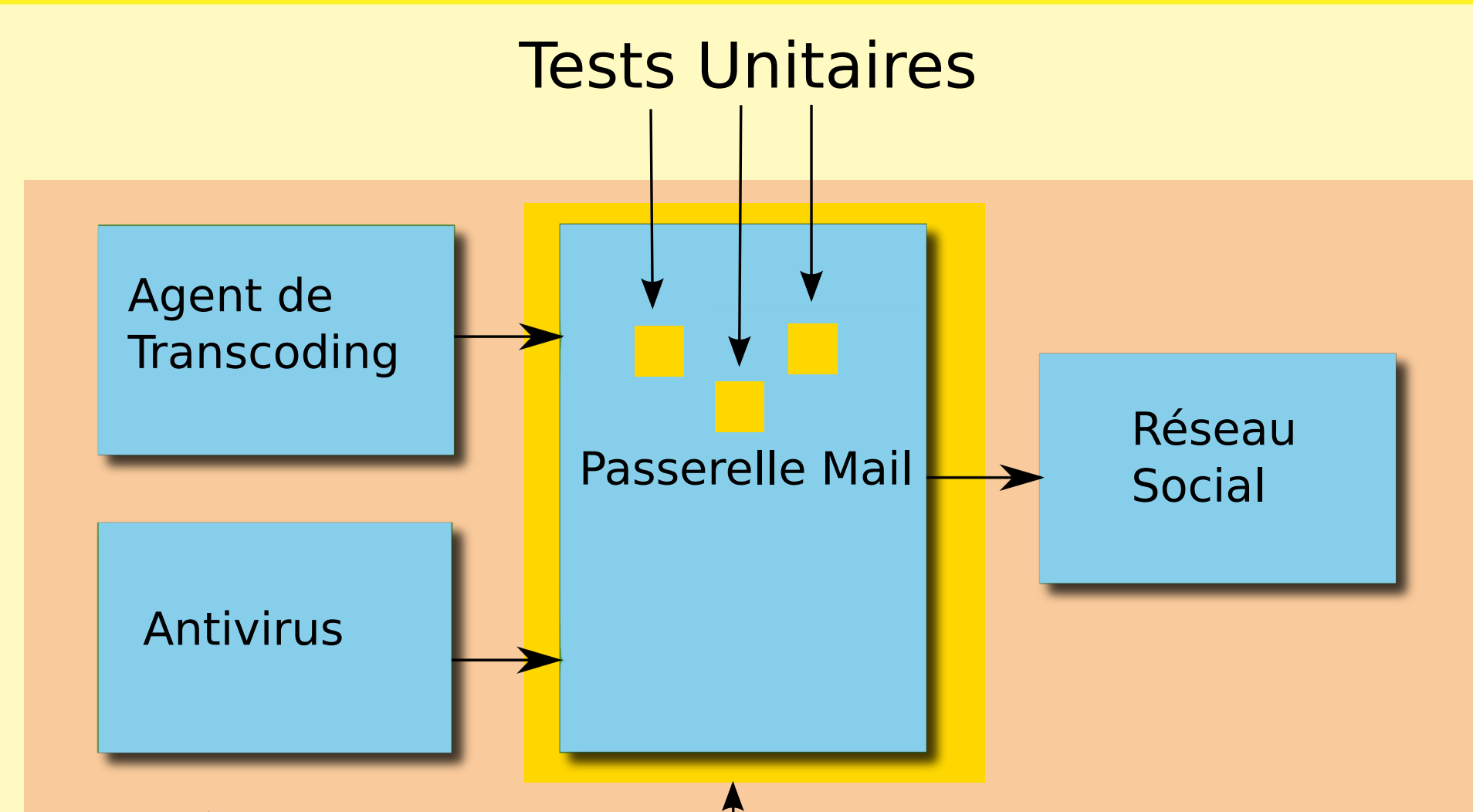
Les domaines sont suffisamment différents pour que les contours métiers soient clairs: *Réseau Social, Transcoding Multimédia, Email, Antivirus*. Nous choisissons de passer sous une architecture Microservices (MS) pour plusieurs raisons:

- Les deux projets doivent être évalués indépendamment sans que l'un compromette l'autre. → **séparation des dépôts de**

sources, pas de code partagé, pas de base de données commune.

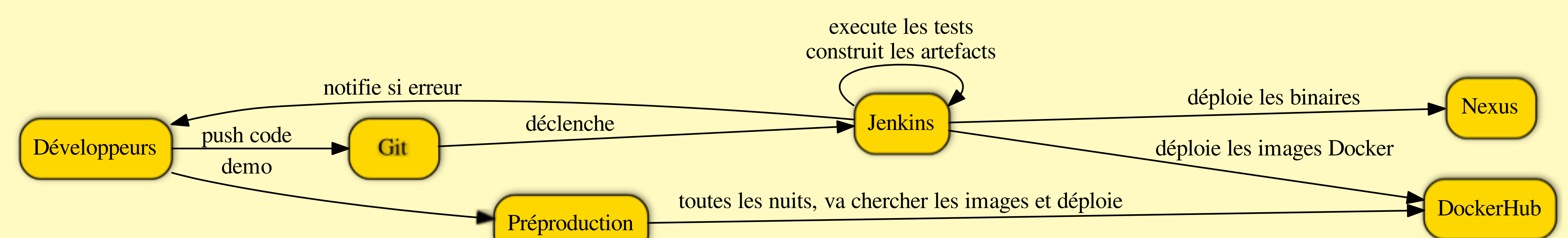
- Communication nécessaire entre services → **Mise en place de l'API REST.**
- Choix du meilleur langage pour chaque service (Java, Python) → **Hétérogénéité des solutions techniques masquée derrière l'API.**
- Nécessité de faire tourner les autres services en local pour développer le sien → **Dockerisation des MS, déploiement des MS pré-configurés dans DockerHub, récupération automatique des images par le dev.**
- Approche agile du développement entraînant une démo du produit global tous les 15 jours → **déploiement de toute la solution avec Docker Compose en pré-prod. La livraison d'un service défectueux peut être reportée.**

Typologie des tests



Trois niveaux de tests automatisés dans une architecture Microservices

Intégration Continue & workflow de développement



Workflow de développement

Tester une architecture Microservice montre certaines particularités par rapport aux monolithes.

- En raison de l'homogénéité du métier, **les tests unitaires** sont plus simples à écrire.
- **Les tests de services** nécessitent l'utilisation de mocks d'interfaces REST, que nous avons réalisé avec Montbank (<http://www.mbttest.org/>).
- Une fois par jour, Jenkins réalise une sé-

rie de **tests de bout en bout** en lançant un environnement iso-production des services et en faisant jouer des scénarios utilisateurs.

- Grâce au DockerHub, toutes les machines autorisées peuvent récupérer les images docker et les lancer à tout moment. Les **bugs sont donc plus facilement reproduits** sur la machine du développeur.

Références

- [1] *Blog Microservice*, de Martin Fowler <http://bit.ly/1dI7ZJQ>
- [2] *Livre Building Microservices Designing Fine-Grained Systems*, Sam Newman, O'Reilly Media 215
- [3] *Podcast Software Engineering Radio*, Episode 213: James Lewis on Microservices, <http://bit.ly/1tkIbeN>

Exemples d'Architectures Microservices

