Livrer chaque jour ce qui est prêt ! Points clés du développement d'un produit avec une livrasion par jour.

Date : 10 avril 2015
Format : Conférence
Speakers : Dimitri Baeli, Benjamin Degerbaix de Les Furets

Contexte du daily delivery

Le comparateur d'assurances LesFurets.com propose 6 produits d'assurance
Le SI des Furets est composé de 2 applications : 1 front et 1 back.
Leur base de code est repose quasi exclusivement sur Java. Les IHM sont codés en GWT et leurs web apps tournent sur Tomcat.
Au total, les Furets c'est 400 000 lignes de code Java, 22 développeurs et 2 architectes

En 2012, lancement de la marque Lesfurets.com.
Part d'une base de code et de process existants :

- Build maven de 15mn
- 200 tests Selenium en 1h
- 12 releases par an avec des Sprints

Depuis janvier 2013, LesFurets.com ont changé d'organisation. Résultats :

- Build maven réduit à 3mn.
- Durée des tests Selenium passé à 10mn (avec le même nombre de tests)
- Livraison de ce qui est prêt à J+1.
- 208 releases du site en 2014 et 47 au 10 avril 2015.

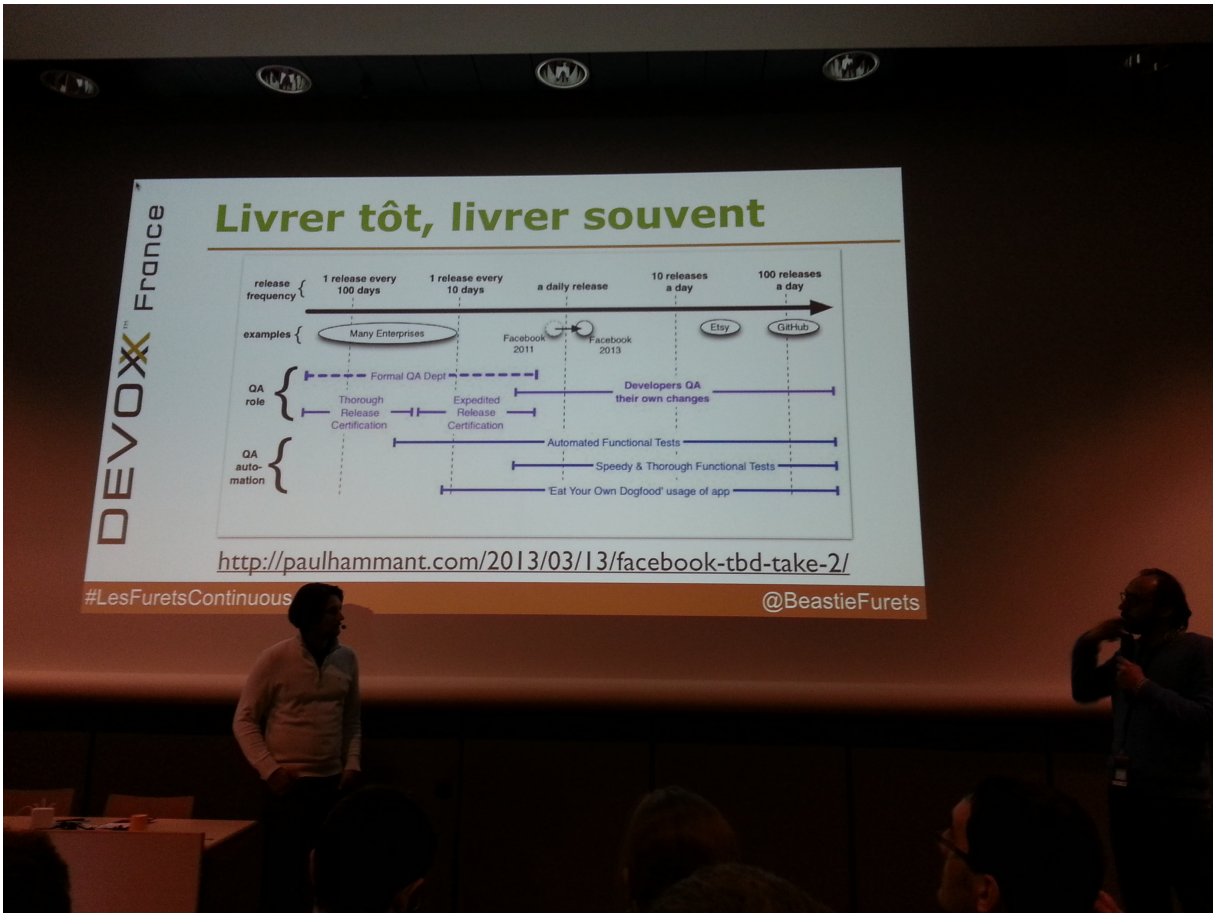
Le Sprint agile s'est transformé en Marathon.
Le temps de passage d'un commit en prod est passé de 2 semaines à 2 jours (c'est ce qu'on appelle le lead time).
Seul ce qui est prêt est livré en prod. L'équipe de LesFurets.com ne raisonne plus avec une roadmap associée à une version.

Points de repère

Ce changement d'organisation a été guidé par différentes sources d'inspiration :

- Manifeste agile : le principe n°1 est la satisfaction du business
- Livrer tôt, livrer souvent : impacte sur l'organisation de l'équipe. Exemple : on ne donne plus de date au métier. Plus besoin d'équipe QA car opportunité de corriger le lendemain. C'est le développeur qui valide ses propres changements.
- Le livre « Continuous Delivery » leur a enseigné :
    - Nécessité d'un build rapide. 15 mn c'était trop long.

    - Build robuste : vert tout le temps
    - Déploiements simples et automatisés : le développeur doit être autonome
    - Monitoring de production et alertes : le daily delivry impose de regarder la production tous les jours. Du monitoring fonctionnel a donc été ajouté.
    - Analyse des causes racines : permet d'éviter qu'un cataclysme revienne.
- Git Flow : son mécanisme de branches les a particulièrement bien aidé. Ils ont adapté à leur besoin la branche develop



Points clés : de la conception à l'exploitation

Amélioration par la fin :

1. Monitoring & Exploitation
2. Mise en production
3. Release Création & Validation
4. Développement
5. Conception

Flux de fonctionnalités :

- Fonctionnalités conçues pour être indépendantes. Les liens entre features sont coupés. Permet de ne pas les livrer en même temps (découplage). Une feature ne

bloque pas les autres. Si malgré ces efforts il subsiste des dépendances entre 2 features, on attend ou on les fusionne
- Fast feedback : des petites tâches vont plus vites en production. Retour chiffré rapide pour prise de décision. Les projets non rentables peuvent être arrêtés très rapidement. Le métier a désormais les outils pour tester leurs idées.

L'environnement de développement :

- Fonctionnement en feature branch : la branche de feature est tirée depuis le master (code de production)
- Environnement quasi iso-prod
- Les développeurs ne sont pas gênés par le travail des autres
- Quid de l'intégration continue avec le feature branch ?
    - Effectué à chaque commit
    - Toutes les features branch sont mergées par l'outil Octopus.

Continous Merge Octopus

Octopus est un outil open source développé par LesFurets ?com et disponible sur leur github.
Chaque feature branch est mergée avec les autres features branch et le master. C'est sur ce commit jetable que l'intégration continue fonctionne. C'est cette branche qui est déployée sur l'environnement de staging. Permet de tester les effets de bord du regroupement, notamment grâce aux tests Selenium.

Release Création & Validation

Création d'une branche de release :

- Entièrement automatisée
- Regroupement des features prêtes (Octopus)
- Déploiement en préprod.

Validation à l'aide d'outils manuels et automatiques pour prévenir les incidents majeurs.
Code review systématique par un autre développeur.
Validation fonctionnelle réalisée sur stage.
Grid Selenium en préprod
Outil de non régression : Zeno Pixel

Grid Selenium on LXC

En série, sur une seule machine, les 200 tests Selenium mettent 6h à s'exécuter.
Sur un Grid Selenium classique, ils mettent 1h.
Avec l'aide de leur Ops, LesFurets.com ont eu l'agréable surprise que le Grid Selenium en RAM (sur RAM FS) ramené le temps d'exécution des 200 tests Selenium à 10 minutes. Axe de performance non documentée.

- Solution peu onéreuse : 1 Machine OVH, 128 Go de RAM, 300 euros / mois
- Activité disque très importante lors du bootstrap du navigateur vu par l'Ops. Les accès disques généraient de l'instabilité dans les tests Selenium. Un des points clés qui les ont permis de partir en prod chaque jour

Zeno Pixel

Les styles CSS en commun peuvent entrainer de graves problèmes.
L'outil Zeno Pixel a été développé en interne. Il se base sur de la comparaison d'images.
Cet outil prend des screenshots automatiques des pages du site sur les différents environnements et les différents devices (desktop, mobile, tablet)
Comparaison des versions de chaque page entre la prod et la pré-prod.
Calcul des différences graphiques.
Test réalisés avant chaque MEP.

Mise en production

Pas de préprosé à la MEP. Chaque jour, un développeur différent effectue la MEP.
Une grande partie automatisée avec Jenkins : création de la release branch, déploiement en prod, merge de la release branche dans le master, création et diffusion de la release note
Possible grâce à l'Infrastructure as Code : config des environnements dans git. Les bugs de configuration sont équivalents à des bugs de code.

Déploiement

Pas de downtime du site. Système Blue / Green.
Environnement de production dupliqué : Version N et Version N-1.
On attend que les sessions http sur l'ancienne instance soient terminés avant de couper tout le trafic (simple configuration de HAProxy).
Utilisation de la même base de données : schéma rétro-compatible.

Git Flow Résumé

- Master : la prod
- 1 branche par feature. Chacune d'elle est créée à partir de la prod. Lorsque la tâche est terminée, elle est mergée dans la branche de pré-prod.
- Branche de pre-prod pour la release. Refusionnée dans le Master.

Les développeurs peuvent être amenés à travailler sur la même feature branch. Si nécessaire, ils peuvent créer d'autres branches à partir de cette feature branch.

Monitoring et Exploitation

Sondes Zabbix – Indicateurs fonctionnels (ex : taux de conversion).
Chaque log d'erreur arrive par mail dans la boite mail du développeur

Post Mortem

- Post Mortem sur chaque incident de Production. Ne pas mettre en prod un jour est considéré comme un incident de prod.