DevOps or not DevOps ?

DevOps & Agile, REX et opportunités

CNES

05/11/2020

There is a better way

Qui sommes-nous ?

Benjamin AMANRICH

Expert DevOps
+33 6 30 97 61 38
b.raphael.amanrich@octo.com

Yohan LASCOMBE

Expert DevOps
+33 6 69 78 17 29
yohan.lascombe@octo.com

L’objectif de cette présentation

01

De l’agilité pour le BUILD uniquement

Équipe avec objectif commun, autonome et responsable

02

Considérer un outil de ticketing comme moyen de communiquer

Les outils ne solutionnent pas les problèmes de communication

03

L’Ops n’intervient qu’au moment de livrer

Intégrer les besoins OPS dès le départ

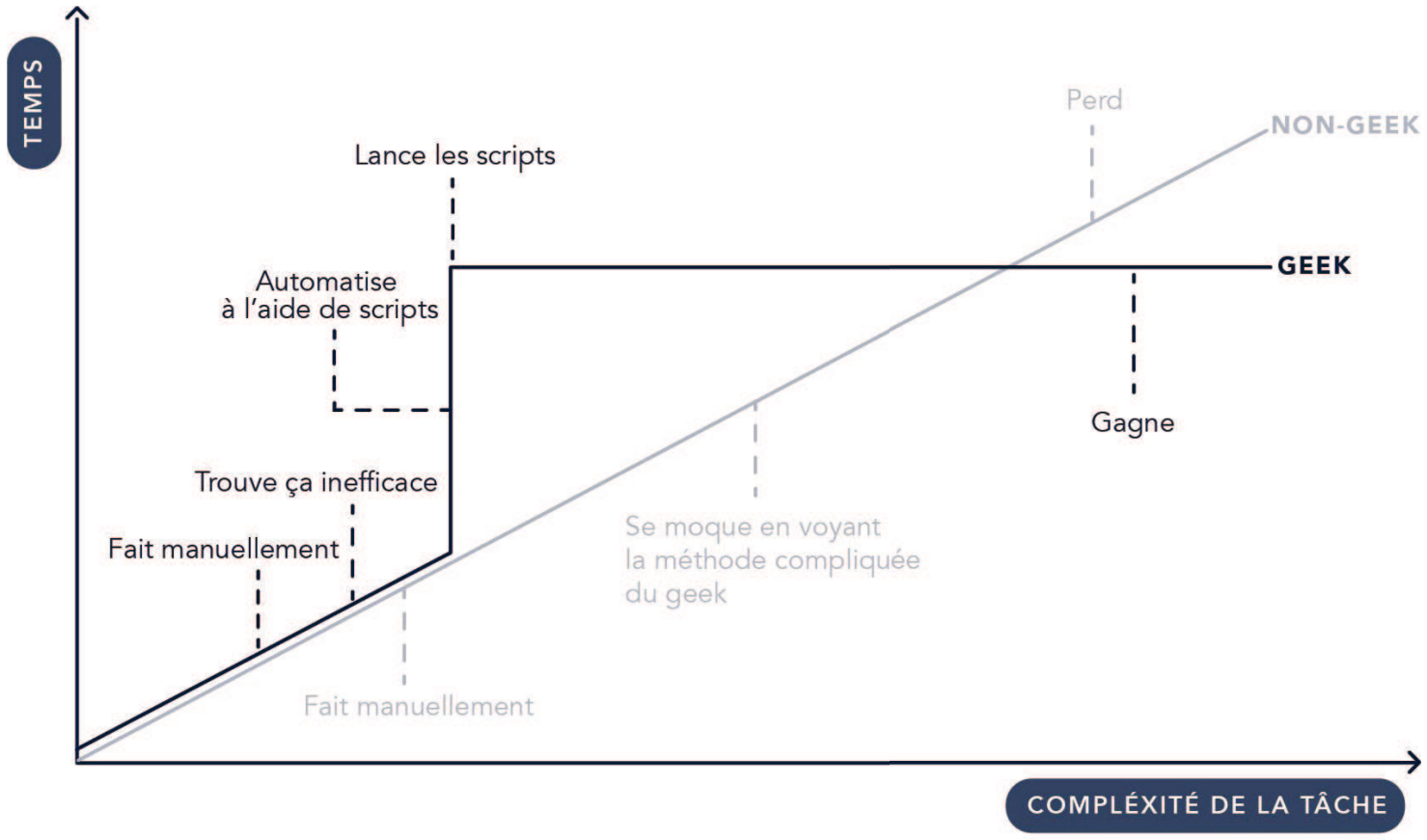
04

Les certitudes avant la mesure

05

Ne pas croire en l'automatisation

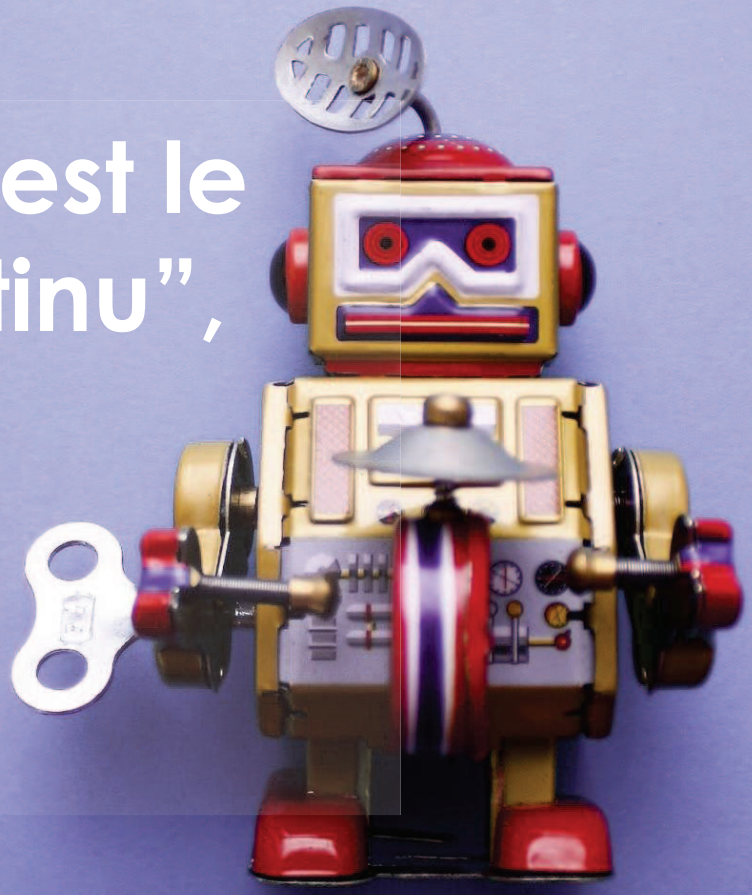
Automatiser au maximum



06

“Cela ne passera jamais la sécurité”

L'enjeu c'est le "tout continu", dev, ops, sécurité, UX, ...



φ DevOps ⇔ Communication

01	De l'agilité pour le build uniquement	Rendre les équipes focus, autonomes et responsables
02	Outil de ticketing pour communiquer	Les outils ne solutionnent pas les problèmes de communication
03	L'Ops n'intervient qu'au moment de livrer	Intégrer les besoins OPS dès le départ
04	Les certitudes avant la mesure	Les mesures permettent de prendre des décisions, les décisions sont challengées par les mesures
05	Ne pas croire en l'automatisation	Automatiser dès la première utilisation
06	"Cela ne passera jamais la sécurité"	L'enjeu c'est le "tout continu", dev, ops, sécurité, UX, ...

Notre objectif

- Nous aidons des clients de tout types (taille différentes, secteur d'activités différents
  - > Dans le cadre de ces missions, nous rencontrons souvent des erreurs similaires
- Choix de vous présenter dans ce talk 6 pièges classiques qu'on rencontre souvent
- Objectif du talk: Détailler ces 6 erreurs classiques et aborder une solution

Fail #1 : De l'agilité pour le BUILD uniquement

Symptômes

- Stock de versions qui s'accumulent
- Découverte problème lors des MEP et livraisons difficiles
- MTTR élevé

Raisons

- Différence de rythme Dev/Ops
- Besoins différents
  - > DEV cherche à livrer des features
  - > OPS cherche la stabilité
- Silotage de l'organisation

Solution

- Mettre en place une équipe autonome et responsable
  - > Une expertise Ops dans l'équipe afin d'y propager l'autonomie et la responsabilité des déploiements
  - > Objectif commun : le produit

Fail #2 : considérer un outil de ticketing comme moyen de communiquer

Symptômes

- Outils de ticketing mis en place
- Des équipe qui éprouvent des difficultés à exprimer leurs besoins
- Des tensions entre les équipes : l'outil n'aide pas

Raisons

- Constat historique d'un problème de communication entre BUILD et RUN
- Processus et outils plutôt que des individus et leur interactions
- Une organisation en silo

Solution

- Favoriser la communication entre les équipes plutôt que mettre en place des process ou des outils
  - > Rituels de communications
  - > Améliorer la communication informelle (BBL, Communautés de pratiques, daily meetings, …)
- Les rituels sont à adapter aux équipes existantes

Fail #3 : L'Ops n'intervient qu'au moment de livrer

Symptômes

- Mise en production complexes et coûteuses
- Tensions entre le BUILD et le RUN
- Des environnements mal maîtrisés

Raisons

- Application qui ne prend pas en compte les problématiques d'opérabilité/d'exploitabilité
- Un design applicatif qui "oublie" que des erreurs peuvent survenir

Solution

- Intégrer les besoins mutuels dès le début des projets
  - > dans le choix d'architectures
  - > dans la démarche de création des environnements
  - > dans les processus CI/CD

Fail #4 : Les certitudes avant la mesure

Symptômes

- ROI des décisions non maîtrisés
- Des priorisations parfois malheureuses
- Tension entre les équipes dans la formalisation des douleurs
- Le système est “dans le noir”

Raisons

- Les douleurs ne sont pas explicités par une mesure : utilisation de ressentis
- Pas de culture de la mesure

Solution

- Mettre en place des indicateurs de performances
  - > Global et non local
  - > Centré sur la capacité produite plus que sur la productivité
  - > Stable et répétable
- Une stratégie d’observabilité systématique
  - > Monitoring
  - > Centralisation des logs
  - > Traçabilité

Fail #5 : Ne pas croire en l'automatisation

Symptômes

- Un TTR élevé
- Perte de capacité à faire lorsque les équipiers partent
- Incidents de mise en production fréquents
- Une communication par la doc

Raisons

- Aucune automatisation
- Des procédures "complètes et à jour" mieux que de l'automatisation
- Erreurs humaines lorsqu'on suit les procédures

Solution

- Systématiser l'automatisation
  - > Infrastructure as Code
- Enrichir le code d'infrastructure avec les pratiques éprouvées par les développeurs:
  - > Outil de gestion de configuration
  - > TDD
  - > Clean code / Clean archi
- Des logiciels opérationnels plus qu'une documentation exhaustive
  - > Générer la documentation **depuis** le code autant que possible
  - > Versionner la documentation supplémentaire

Fail #6 : “Cela ne passera jamais la sécurité”

Symptômes

- Blocage de la mise en production par la sécurité
- Architecture chamboulée au dernier moment
- Un produit qu ne trouve pas ses utilisateurs

Raisons

- Application qui ne tient pas compte des contraintes de sécurité
- Equipe pas ou trop peu accompagnée par les experts sécurité

Solution

- DevOps → DevSecOps
  - > On casse le mur entre la sécurité et le DevOps
- DevOps → Dev ‘*’ Ops
  - > Securité
  - > UX/UI
  - > Performances
  - > ...