302 - La culture DevOps

WIK-DEVOPS-302

Intervenant : Jeremy Trufier <jeremy@wikodit.fr>

ynov

wikodit

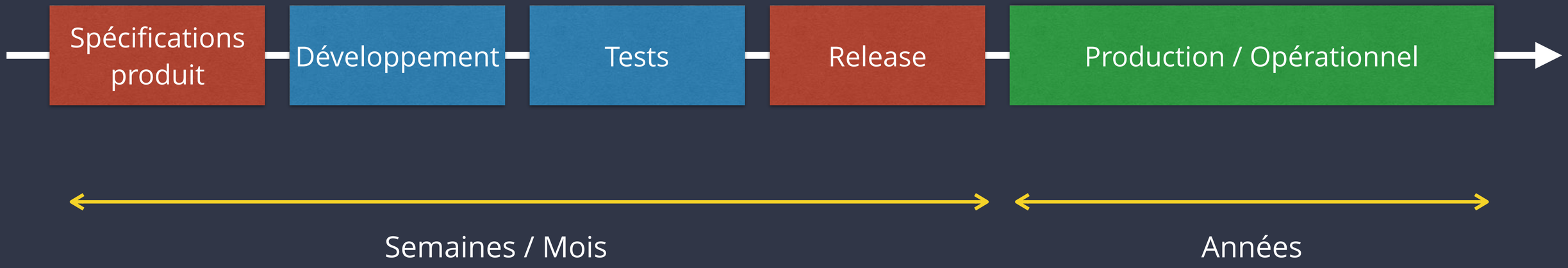
WIK-DEVOPS

Programme DEVOPS

301 – Introduction, Kaizen et CAMS	1 séance
302 – La culture DevOps	1 séance
303 – Automatisation / Déploiement continu	4 séance
304 – Mesure & Monitoring	2 séance
305 – Partage	2 séance

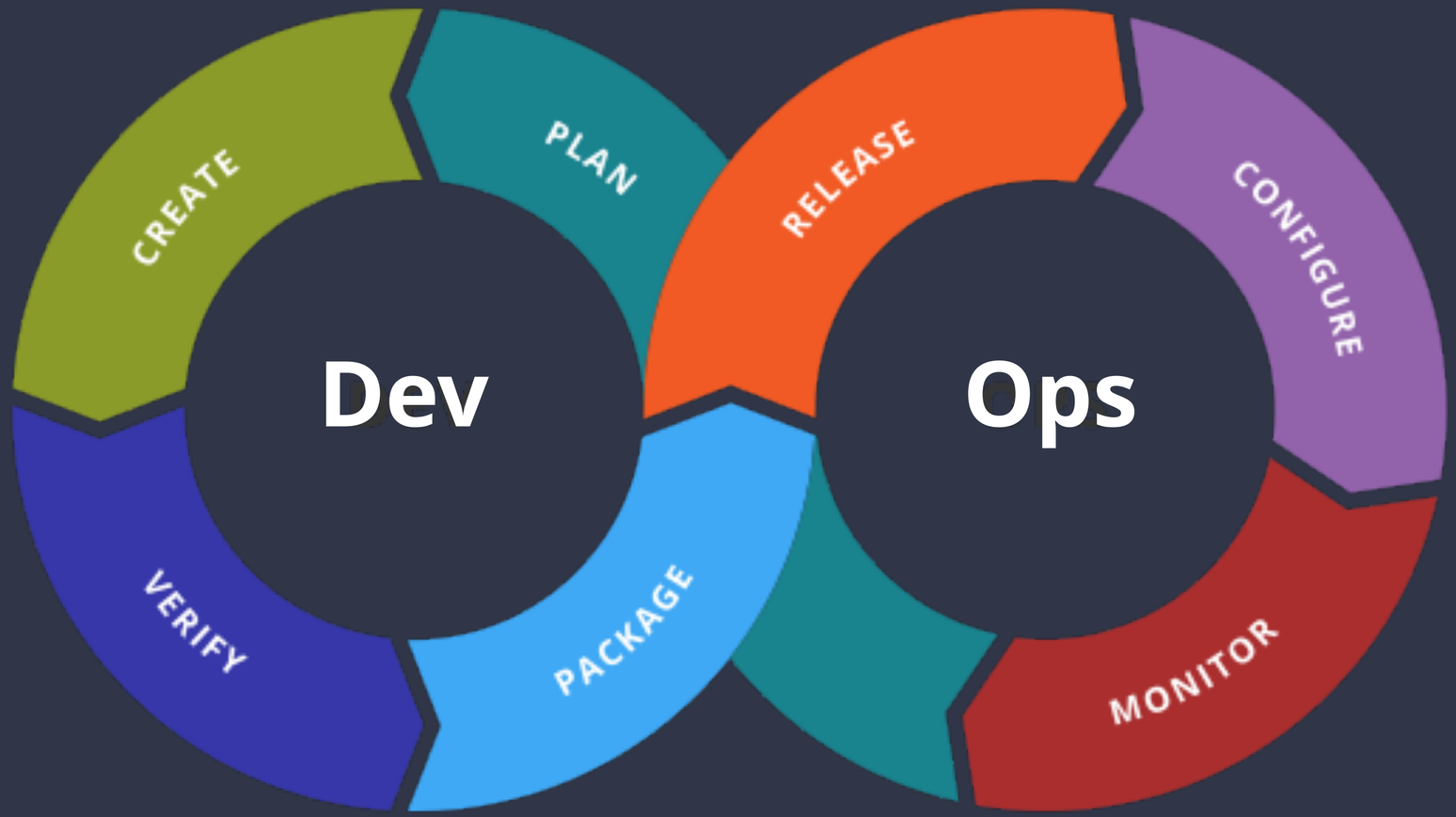
I. Cycle de vie du produit et DevOps

Cycle de vie **Traditionnel**



Qu'est ce qui génère de l'argent ?

Cycle de vie DevOps



- Meilleur Flow
- Feedback rapide
- Déploiement petits et réguliers
- Retour monétaire immédiats
- Amélioration

II. Service Ownership

You wrote it, you run it !

- Responsable du développement
- Responsable du déploiement
- Responsable de l'opérationnel
- Responsable de l'instrumentalisation
- Mais responsable des problèmes !

Chez Google

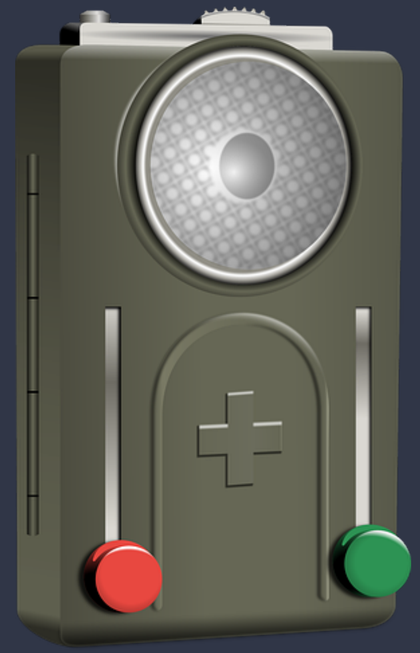
- SRE Team : Service Reliability Engineer
- Définition d'un budget min SLA pour les Dev
- 50% en Ops, le reste en dev
- Healthy / Unhealthy
- SRE gère le feu vert pour le déploiement de nouvelle features

Feature toggle

- Activer / Désactiver des fonctionnalités au choix
- Ajout d'un flag sur les features
- Possibilité d'activer les features seulement pour certains groupes de personnes
- Ou seulement pour 10% des visiteurs
- Les Ops ont la main en cas de problème

=> Minimisation des risques

=> DarkLaunches (Facebook, Google)



III. Dette technique

Du code = de la dette

- "Pas touche à ça, sinon ça casse tout"
- Code dupliqué
- Code sur lequel une seule personne peut intervenir
- Bottlenecks de performance sur certains composants

Dette volontaire

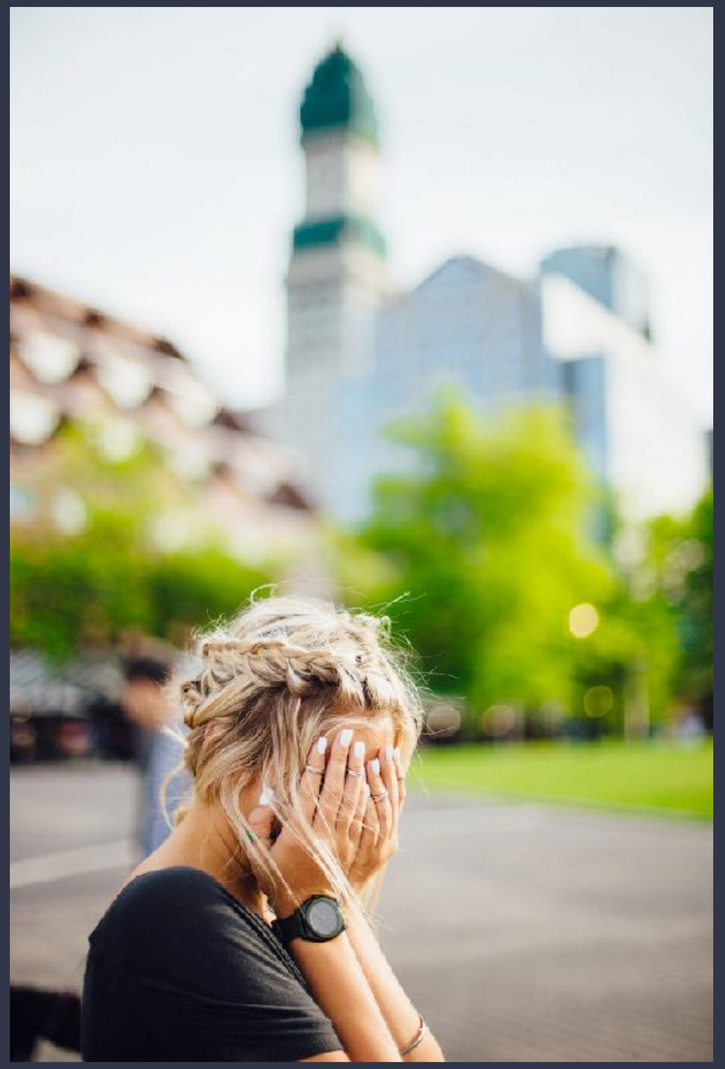
- Décisions de dernière minute
- Mauvais choix
- Se tromper sans rectifier
- Pas de tests automatisés

Dette involontaire

- Trop d'évolutions, pas de refactoring
- Mauvaise architecture
- Composant qui tombe dans l'oubli

Et alors ?

- Il faut toujours rembourser ses dettes !
- mais : "Si ça fonctionne, on ne touche pas !"
- pourtant : Risque élevé
- Développement qui devient de plus en plus couteux



Comment la résoudre

- Admettre la dette
- Planifier sur le long terme (morceau par morceau)
- Utiliser les tests automatisés
- Attention de ne pas arrêter le flux de production !

IV. Destructive testing

Mon produit est-il prêt ?

- Prêt pour la production ?
- Release Checklist :
  - Besoins serveurs
  - Tests automatisés
  - Tests destructives
  - Feedback des béta-testeurs

Du monde industriel

- Crash test
- Stress test
- Hardness test
- IP tests (indice de protection)

Pour les Ops

- Simulation d'accident sur serveurs
- Simulation coupure de courant, d'internet
- Simulation conditions naturelles
- Stress-test requêtes
- Simulation de packets lost

Pour les devs

- Déploiement annulé
- Pen-test (tests d'intrusions) & Hack
- Stress-tests

Comment ?

- Prévoir un moment, ex: 1h le jour où le service est le moins utilisé
- Réunir tout le monde
- Se concentrer sur un composant
- Risqué, mais le risque est contrôlé

Le vendredi tout est permis !

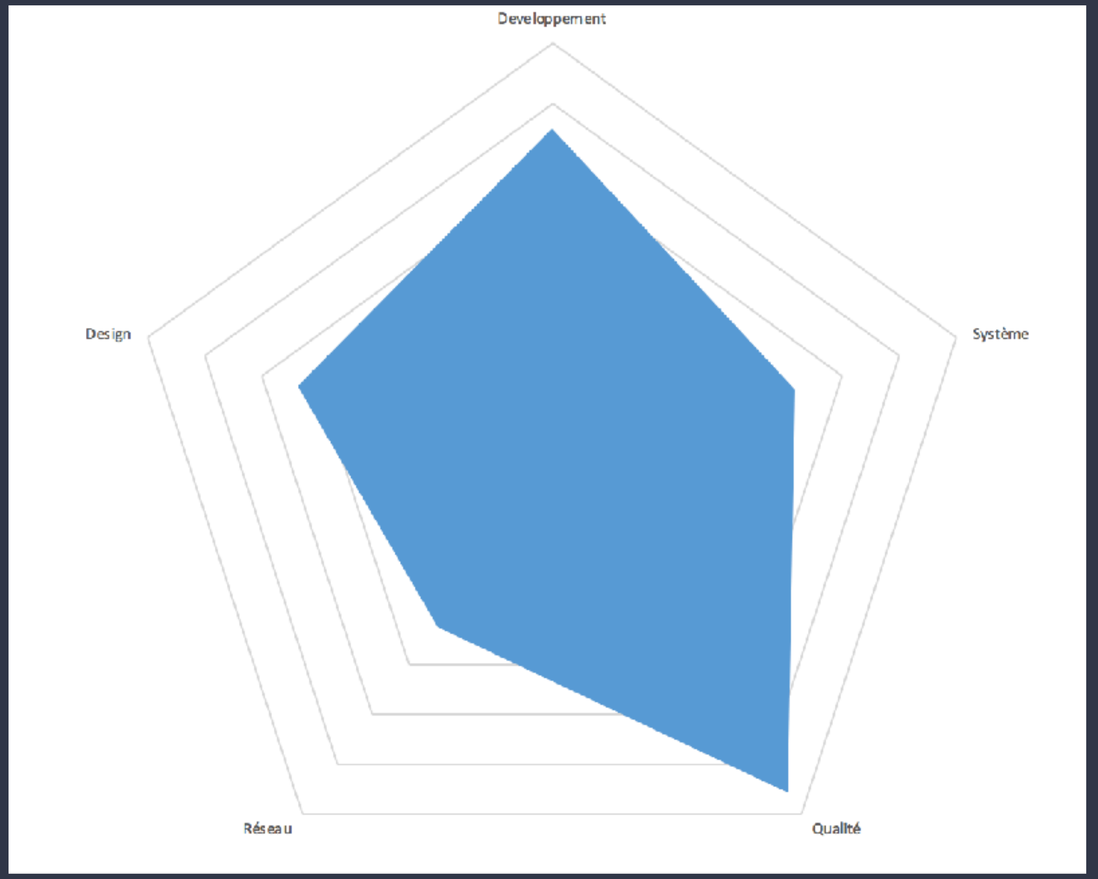
V. Équipes multi-fonctions

Un même but

- Composition variée : Système, Qualité, Support, Développement, Designer, ...
- Concentré sur un produit, ou sur une partie spécifique d'un produit
- Un même but = même longueur d'onde = meilleur Flow

Ensemble


Paul
Yannick
Lucie
Marc
Émilie
Developpement
Système
Qualité
Réseau
Design


Developpement
Système
Qualité
Réseau
Design

Partage de connaissance

- Tout le monde évolue ensemble
- Les compétences se partagent au fur à mesure
- Chacun devient capable de prendre les parties des autres
- Absences facilités
- Intégration des nouveaux facilité

Félicitations !!

Cours WIK-DEVOPS-302 burned :)