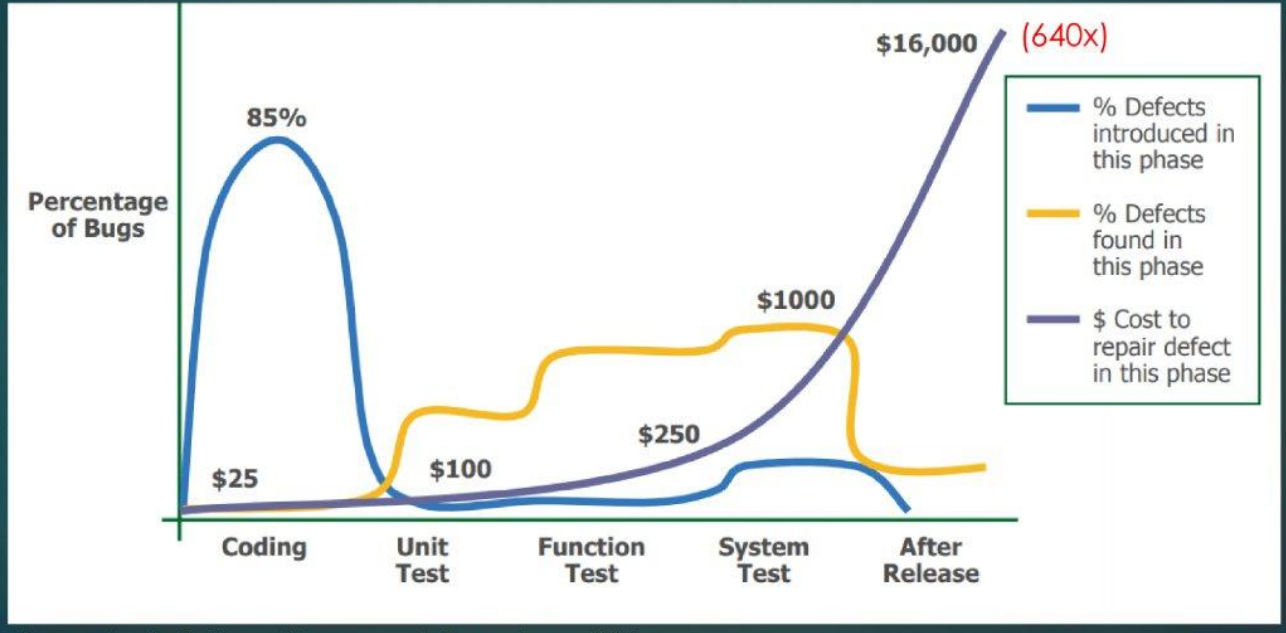
Besoin en intégration et déploiement continu

CI/CD: Pour quoi faire ?

Sébastien Valat - INRIA
01/07/2024

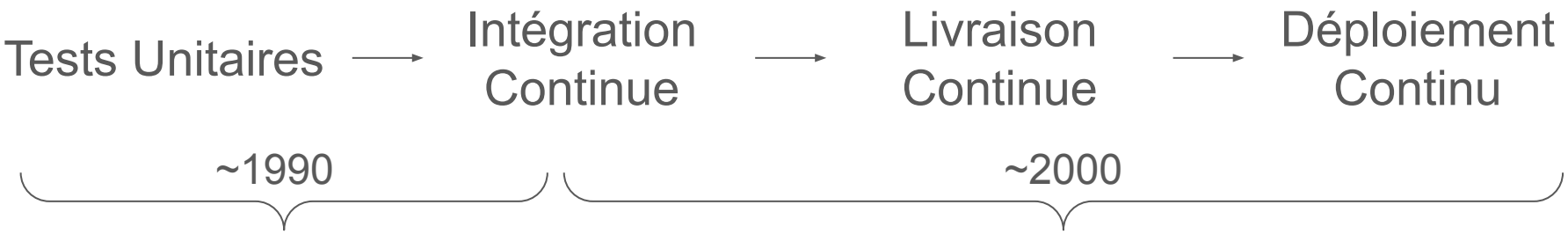
Came back to software….

Capers Jones, 1996



Source: Applied Software Measurement, Capers Jones, 1996

CI/CD: Automatiser pour augmenter la productivité



↗️ Productivité des développeurs (Quantité ET Qualité)

↗️ Productivité des “DevOps” Réactivité des opérateurs

- À l’origine de technologies adaptées / en tirant parti : **Conteneurs**
- Indispensable pour le **cloud** : déploiements massifs d’applications hétérogènes
- À l’origine de nouvelles méthodes de management des équipes et de l’infrastructure: **infrastructure as code**, **méthodes agiles**

Cas d’usage: Publication

- Scénario : Héberger et générer l’article à partir des résultats
- Gains :
  - Productivité (en particulier si collaboration)
  - Reproductibilité (“Papers with code”)
  - Souveraineté
- Pratique fortement encouragée : école S3
  - https://github.com/s3-school/pkoffee-solution
  - https://s3-school.github.io/pkoffee-solution/analysis.html
- Gitlab
  - Héberge le code de l’analyse et de l’article (y compris figures)
  - CI : génération des figures et de l’article
  - CD : héberge l’article rendu (pdf) et la documentation associée

Cas d’usage: Développement Logiciel - Prototype

- Scénario : Développement collaboratif par 1 équipe
- Gains :
  - Productivité développeurs et utilisateurs (déploiement)
  - Visibilité et transparence
  - Souveraineté
- Exemple : gammalearn
  - CI: Unit tests, linting, documentation
  - CD: packaging noarch (python), conteneurisation, release manuelle
- Gitlab
  - Héberge le code
  - Schedule le computing (runner LAPP) pour CI/CD
  - Héberge les artéfacts : paquets, documentation, conteneurs

Cas d’usage: Développement Logiciel - Plus poussé

- Scénario : Développement collaboratif, multiples contributeurs et utilisateurs
- Gains :
  - Productivité développeurs et utilisateurs (déploiement)
  - Visibilité et transparence
  - Souveraineté
- Example: Phoenix2, ~100 projets d’un écosystème cohérent
  - CI : Unit tests, linting, documentation, MAJ dépendances
  - CD : packaging multi-platform, multi-language, conteneurisation, releases automatiques
- Gitlab
  - Héberge le code, centralise la définition des CI/CD
  - Schedule le computing (runner externe) pour CI/CD
  - Héberge les artéfacts : paquets, documentation, conteneurs, release notes…
  - À venir : déploiement pour benchmarks automatiques ?

CI/CD: besoins et évolution

- **Indispensable** pour la productivité des équipes (quantité ET qualité)
  - Automatisation : échanger du temps humain contre du computing
  - Rentable, démontré par les évolutions de l'informatique (cloud, DevOps) depuis 20 ans
  - Particulièrement pertinent quand les ressources humaines sont limitées ?
- **Forge logicielle centrale** dans le processus : **outil critique**
  - Hébergement du code, scheduling CI/CD → utilisation **constante** par nos devs
  - Hébergement des artéfacts → utilisation par nos **partenaires**
- **Sous-utilisé** à l'IN2P3 → va (doit) croître fortement
  - Pas assez utilisé par les chercheurs pour les publications
  - Utilisation sous-optimale par les informaticiens : pas d'expertise partagée dans les labos
  - Utilisation par d'autres corps de métier : électroniciens ?
- Ressources requises pour le GitLab, outil critique **actuellement sous-dimensionné**
  - Instance centrale : doit tenir la charge pour tous les utilisateurs
    - Collaboration indispensable → compte EduGain ?
  - CI computing: peut-être distribué aux labos / expériences
    - Projets transverses ? Développement cross-expérience ? Formation ? R&D ?
    - Expériences qui refusent la CI/CD → nos devs ralentis ?
    - Duplication de la charge pour mettre en place / maintenir ces infrastructures ?