

# DevOps

## Introduction

Thomas Ropars

`thomas.ropars@univ-grenoble-alpes.fr`

`https://tropars.github.io/`

2026

# Présentation

## Organisation du cours

- 12 heures de cours / 21 heures de TP
  - ▶ Un projet en groupes sur 9 heures
- Utilisation de Moodle pour les TPs
- Slides de cours disponibles sur:  
<https://tropars.github.io/teaching/>

## Évaluation

- Note de TP (40%)
  - ▶ Note de projet
  - ▶ + prise en compte du travail en TP
- Examen final (60%)

# Staff

- Thomas Ropars:  
`thomas.ropars@univ-grenoble-alpes.fr`
- Danilo Carastan-Santos:  
`danilo.carastan-dos-santos@univ-grenoble-alpes.fr`
- Mohammed Khatiri: `dr.khatiri.med@gmail.com`

# Objectifs du cours

## Qualité du logiciel

- Des méthodes
- Des outils

## Approche DevOps

- Intégration continue
- Livraison continue
- (Déploiement continu)

# Contexte

- Systèmes informatiques
  - ▶ 80% de logiciel
  - ▶ 20% de matériel
- Le matériel est fourni pour un nombre restreint de fabricants
  - ▶ Peut être considéré comme relativement fiable
- La plupart des fonctionnalités dans les systèmes informatiques sont fournies par le logiciel

# Enjeux

## Standish Group 2015 CHAOS report

Analyse de plus de 50000 projets de développement logiciel à travers le monde.

### Réussite des projets

- Succès<sup>1</sup>: 29%
- Problématiques: 52%
- Échec: 19%

16% de succès en 1994.

---

<sup>1</sup>Succès = projet terminé dans les temps, sans dépassement de budget

# Enjeux

## Standish Group 2015 CHAOS report

La probabilité de succès décroît avec la taille du projet:

- Petit<sup>1</sup>: 70% de succès
- Moyen: 22% de succès
- Grand<sup>2</sup>: 11% de succès

Attention les chiffres de ce rapport sont parfois contestés. Ils donnent tout de même une idée des enjeux.

---

<sup>1</sup>Petit = moins de 1M\$

<sup>2</sup>Grand = plus de 6M\$

# Causes des défaillances pour applications clouds

Analysis of Business Data Processing Cloud Apps (Di Martino et al (2012))

- 34% des défaillances sont dues à des entrées utilisateur non prévues
- 32% des défaillances sont des timeouts
- 95% des défaillances sont dues aux mêmes 5 modules (37% des LOCs)



# Causes des défaillances pour applications clouds

Analysis of Business Data Processing Cloud Apps (Di Martino et al (2012))

- 34% des défaillances sont dues à des entrées utilisateur non prévues
- 32% des défaillances sont des timeouts
- 95% des défaillances sont dues aux mêmes 5 modules (37% des LOCs)

Enseignements:

- Il faut traiter correctement les exceptions
- Un petit nombre d'erreurs sont à l'origine de la majorité des défaillances → De meilleures procédures de tests sont requises

## D'autres défaillances dans les nuages

- Amazon Web Services (2015)
  - ▶ Indisponibilité de 6 heures
  - ▶ Source: problème dans la gestion des méta-données dans DynamoDB (base de donnée NoSQL)
  - ▶ La défaillance s'est propagée à d'autres services
  - ▶ 2 des plus gros clients de AWS sont Amazon.com et Netflix
    - Netflix a annoncé ne pas avoir été impacté
    - Une analyse des plaintes des consommateurs montrent une très forte hausse pendant cette période

## Coût d'une indisponibilité dans le cloud

- Amazon.com a subi une panne de 45 minutes en 2013
- Le coût en terme de ventes a été estimé à 4 millions de dollars

# Coût du développement logiciel

Source E. Chenu

## Ordres de grandeur

- 1 H/An = 1350 heures
- 1h  $\simeq$  50 €
- Productivité  $\simeq$  2 à 5 lignes/h

Dimension	Nb lignes	Heures	Coût	Hommes/An
Petit	30000	6000	300 K€	4
Gros	500000	100000	5 M€	74

# Génie Logiciel

## Une Définition

Le terme génie logiciel désigne l'ensemble des méthodes, des techniques et outils concourant à la production d'un logiciel, au delà de la seule activité de programmation

# Génie Logiciel

## Une Définition

Le terme génie logiciel désigne l'ensemble des **méthodes**, des **techniques** et **outils** concourant à la production d'un logiciel, au delà de la seule activité de programmation

# Les défis

- La taille des projets: pour certains, des millions de ligne de code (MLOC)
- Contribution à des projets existants:
  - ▶ Peu de projets démarrés *from scratch*
  - ▶ Plus de valeur d'ajouter 100 LOCs à un grand projet largement utilisé que d'écrire 10000 LOCs dans son coin.
- Réutilisation de code existant
  - ▶ Est ce que le concepteur d'une voiture commence par réinventer la roue?
- Collaboration avec d'autres développeurs
  - ▶ Comment interagir?
  - ▶ Comment fournir du code réutilisable?
- Des utilisateurs/clients

# Les enjeux

L'industrie du logiciel, c'est 5% de projets "*from scratch*" et 95% de projets existants. Le travail consiste alors à:

- Réutiliser
- Faire évoluer
- Étendre
- Adapter
- Maintenir
- Réorganiser

# DevOps

## Définition (simple)

Ensemble de techniques et d'outils facilitant le passage du développement à la production.



## Définition (simple)

Ensemble de techniques et d'outils facilitant le passage du développement à la production.

## Bien plus que ça:

- Modèle de fonctionnement de l'entreprise
  - ▶ Impliquant tous les maillons de la chaîne (RHs, finances, etc.)
- Modèle d'interactions entre les équipes
- Intégration du retour sur expérience
- Une “culture”

## Définition (simple)

Ensemble de techniques et d'outils facilitant le passage du développement à la production.

## Bien plus que ça:

- Modèle de fonctionnement de l'entreprise
  - ▶ Impliquant tous les maillons de la chaîne (RHs, finances, etc.)
- Modèle d'interactions entre les équipes
- Intégration du retour sur expérience
- Une “culture”

**Nous en resterons à la définition simple**

# DevOps

Relation entre **Dev** et **Ops**:

- **Dev**: Équipes de développeurs logiciels
- **Ops**: Équipes en charge de la mise en production des produits

Antagonisme fort:

- **Dev**: Modifications aux moindres coûts, le plus rapidement possible
- **Ops**: Stabilité du système, qualité

**L'automatisation** est au cœur de l'approche DevOps

# DevOps: Automatisation

## Intégration continue

Une méthode de développement logiciel dans laquelle le logiciel est reconstruit et testé à chaque modification apportée par un programmeur.

## Livraison continue

La livraison continue est une approche dans laquelle l'intégration continue associée à des techniques de déploiement automatiques assurent une mise en production rapide et fiable du logiciel.

## Déploiement continu

Le déploiement continu est une approche dans laquelle chaque modification apportée par un programmeur passe automatiquement toute la chaîne allant des tests à la mise en production. Il n'y a plus d'intervention humaine.

# DevOps: Un mouvement de fond

## Les grands acteurs de production logiciel

- Google
  - Netflix
  - Mozilla
- 
- Adoptée par une part importante des entreprises qui font du dev. logiciel

# Etat actuel

Rapport DORA "state of DevOps" – 2023

## Etude au près de 36 000 professionnels

- Les 2 technique les plus importantes mises en oeuvre par les équipes de développement logiciel:
  1. Intégration continue
  2. Livraison continue
- Un aspect critique de la performance d'une entreprise est la performance pour livrer de nouveaux logiciels. Les métriques prises en compte sont:

# Etat actuel

## Rapport DORA "state of DevOps" – 2023

### Etude au près de 36 000 professionnels

- Les 2 technique les plus importantes mises en oeuvre par les équipes de développement logiciel:
  1. Intégration continue
  2. Livraison continue
- Un aspect critique de la performance d'une entreprise est la performance pour livrer de nouveaux logiciels. Les métriques prises en compte sont:
  - ▶ Fréquence des déploiements
  - ▶ Délai de mise en production
  - ▶ Taux de déploiements problématiques
  - ▶ Temps nécessaire à la résolution des problèmes

# Un exemple extrême de tests: The Netflix Simian Army

<http://techblog.netflix.com/2011/07/netflix-simian-army.html>

## Contexte

- Netflix utilise Amazon AWS comme fournisseur de ressources informatiques
  - ▶ Netflix ne gère pas ses propres ressources matérielles
- La disponibilité de ses services est fondamentale
  - ▶ Les clients de Netflix ne peuvent accepter des interruptions de service fréquentes
- Netflix a peu de contrôle sur la fiabilité des ressources qui lui sont fournies par Amazon
  - ▶ La disponibilité doit être assurée au niveau logiciel



# The Netflix Simian Army

Chaos Engineering:

<https://www.usenix.org/conference/lisa18/presentation/jones>

**The best way to avoid failure is to fail constantly**

# The Netflix Simian Army

Chaos Engineering:

<https://www.usenix.org/conference/lisa18/presentation/jones>

**The best way to avoid failure is to fail constantly**

Une armée de “singes testeurs” :

- **The Chaos Monkey**: De manière régulière tue un des serveurs de Netflix
- **The Latency Monkey**: Augmente artificiellement la latence entre client et serveur
- **The Chaos Gorilla**: Simule la disparition de tous les serveurs dans une région du monde
- etc.

# Impact du DevOps sur le travail d'un développeur logiciel

[https:](https://www.usenix.org/conference/lisa17/conference-program/presentation/legat)

[//www.usenix.org/conference/lisa17/conference-program/presentation/legat](https://www.usenix.org/conference/lisa17/conference-program/presentation/legat)

Organisation du temps de travail:

## Vision classique

- 60% de programmation
- 20% de tests
- 10% d'intégration
- 10% de documentation

## DevOps

# Impact du DevOps sur le travail d'un développeur logiciel

[https:](https://www.usenix.org/conference/lisa17/conference-program/presentation/leg)

[//www.usenix.org/conference/lisa17/conference-program/presentation/leg](https://www.usenix.org/conference/lisa17/conference-program/presentation/leg)

Organisation du temps de travail:

## Vision classique

- 60% de programmation
- 20% de tests
- 10% d'intégration
- 10% de documentation

## DevOps

- 45% de programmation
- 10% de tests
- 10% d'intégration
- 15% de documentation

# Impact du DevOps sur le travail d'un développeur logiciel

[https:](https://www.usenix.org/conference/lisa17/conference-program/presentation/legat)

[//www.usenix.org/conference/lisa17/conference-program/presentation/legat](https://www.usenix.org/conference/lisa17/conference-program/presentation/legat)

Organisation du temps de travail:

## Vision classique

- 60% de programmation
- 20% de tests
- 10% d'intégration
- 10% de documentation

## DevOps

- 45% de programmation
- 10% de tests
- 10% d'intégration
- 15% de documentation
- 10% automatisation des tests
- 10% automatisation du déploiement

# Mots clés du cours

- Gestionnaire de versions
- Versions
- Source code management
- Automatisation
- Débugger
- Tests unitaires
- Couverture de code
- Workflow Git
- Revue de code
- Intégration continue
- Livraison continue
- Conteneurs