FONDAMENTAUX DEVOPS

ORIGINE DU TERME

Le terme "DevOps" est né de la fusion des mots "**Development**" (Développement) et "**Operations**" (Opérations).

ORIGINE DU TERME

Il fait référence à la collaboration entre les équipes de développement et d'exploitation pour **améliorer les processus de livraison de logiciels**.

RÔLE

Le but du DevOps est d'optimiser le cycle de vie des applications en facilitant la communication et la coopération entre les différentes équipes impliquées.

OBJECTIFS

- **Améliorer la communication** et la **collaboration** entre les équipes
- **Automatiser** les tâches répétitives et les **processus de livraison**
- Favoriser l'**amélioration continue** et la **qualité** des produits logiciels

AVANTAGES

- **Livraison plus rapide** des fonctionnalités
- Moins de problèmes dans la **production**
- Temps de **rétablissement réduit** en cas d'incidents
- **Meilleure satisfaction** des clients et des utilisateurs

CULTURE ET PHILOSOPHIE DEVOPS

COLLABORATION ENTRE ÉQUIPES

Dans un environnement **DevOps**, les équipes de **développement**, d'**exploitation** et d'autres parties prenantes travaillent en étroite collaboration pour assurer une **livraison rapide**, **fiable** et de **qualité** des logiciels.

AMÉLIORATION CONTINUE

Les équipes utilisent des **métriques** et des **outils de surveillance** pour identifier les problèmes et les résoudre rapidement, afin d'optimiser les performances et la fiabilité des applications.

APPRENTISSAGE ET PARTAGE

Le **partage** des pratiques, des **connaissances** et des **expériences** entre les équipes permet d'améliorer les compétences et la performance de l'ensemble de l'organisation.

APPRENTISSAGE ET PARTAGE

Les organisations DevOps encouragent l'apprentissage et le partage pour renforcer l'**amélioration continue**.

LES PRINCIPES DU DEVOPS

Les principes fondamentaux du **DevOps** sont basés sur la **collaboration**, l'**automatisation** et l'**amélioration continue** dans le but d'accélérer le développement et la livraison de logiciels de qualité.

CALMS

CALMS est un **acronyme** décrivant les principaux éléments pour la mise en œuvre réussie du **DevOps**.

CALMS

Acronyme	Description
C	Culture
A	Automation
L	Lean
M	Measurement
S	Sharing

CULTURE

Le **DevOps** favorise une culture de **collaboration** et de **communication** entre les équipes de développement, d'exploitation et de support.

AUTOMATION

Automatiser les processus de **développement**, de **test** et de **déploiement** pour réduire les erreurs humaines et accélérer la livraison de logiciels.

AUTOMATION

- Gestion du code source
- Intégration continue
- Tests automatisés
- Déploiement continu

LEAN

Adopter des pratiques de gestion **Lean** pour éliminer les **gaspillages**, réduire les **délais** et maximiser la **valeur** pour les clients.

LEAN

Les 7 gaspillages dans le Lean :

- Surproduction
- Temps d'attente
- Transport
- Surstockage
- Surtraitement
- Mouvements inutiles
- Défauts

MEASUREMENT

Mesurer les **performances** et les **résultats** pour prendre des décisions éclairées et identifier les domaines d'amélioration.

SHARING

Partage des **connaissances**, des **expériences** et des **meilleures pratiques** pour favoriser l'apprentissage et l'**amélioration continue**.

LES TROIS PILIERS DU DEVOPS

Les trois piliers du **DevOps** sont les **infrastructures**, l'**automatisation** et la **collaboration**.

PILIERS

Pilier	Description
Infrastructures	Gestion des ressources matérielles et logicielles pour soutenir les applications et les services
Automatisation	Mise en place des processus pour faciliter les déploiements et les mises à jour automatiques
Collaboration	Communication et coopération entre les équipes de développement et d'exploitation pour les aligner

INFRASTRUCTURES

Mise en place d'**infrastructures flexibles** et **évolutives** pour soutenir les processus de **développement**, de **test** et de **déploiement**.

AUTOMATISATION

Automatiser les processus manuels pour **réduire** les erreurs, **accélérer** la livraison de logiciels et **améliorer** la qualité.

AUTOMATISATION

- Réduction des erreurs humaines
- Accélération de la livraison de logiciels
- Amélioration de la qualité des produits

COLLABORATION

Encourager la **collaboration** et la **communication** entre les équipes pour développer un environnement de travail **transparent** et **efficace**.

OUTILS DE COLLABORATION

Avantages de la collaboration	Outils couramment utilisés
Partage de connaissances	Slack, Microsoft Teams
Réduction des silos	Jira, Trello
Meilleure automatisation	GitHub, GitLab
Processus de développement plus rapide	Jenkins, Docker

COMPOSANTS CLÉS DU DEVOPS

INFRASTRUCTURE AS CODE (IAC)

L'**Infrastructure as Code (IaC)** est la pratique qui consiste à gérer et à **provisionner** les ressources informatiques à l'aide de **fichiers de configuration**.

PRINCIPES

- Utilisation de fichiers de configuration pour définir les ressources
- Permet la réplication facile d'infrastructures
- Favorise l'utilisation des meilleures pratiques et l'automatisation

DÉFINITION

Infrastructure as Code (IaC) est une approche pour définir, gérer et automatiser les infrastructures informatiques en utilisant des fichiers de configuration lisibles par l'homme et des scripts automatisés.

EXEMPLES D'OUTILS

TERRAFORM

Outil open-source d'Infrastructure as Code (IaC) permettant de définir et fournir des infrastructures dans les principaux fournisseurs de cloud.

ANSIBLE

Outil open-source d'automatisation, de déploiement et d'orchestration utilisé pour gérer la configuration des serveurs.

ARM TEMPLATES

Modèles de déploiement pour automatiser la création et la gestion des ressources Azure.

AVANTAGES

- **Automatisation**
- **Traçabilité** et contrôle des versions
- Réduction des **erreurs humaines**
- Facilite la **collaboration**

INTÉGRATION CONTINUE (CI)

L'**intégration continue (CI)** est une pratique de développement visant à fusionner les modifications de code dans un **référentiel centralisé** et à exécuter des **tests automatisés** pour détecter les problèmes le plus tôt possible.

DÉFINITION

Intégration continue est une pratique de **développement logiciel** qui vise à intégrer régulièrement **les modifications de code** dans un référentiel centralisé et à exécuter des **tests automatisés** pour détecter rapidement les problèmes et les résoudre.

EXEMPLES D'OUTILS

- Jenkins
- Travis CI
- GitLab CI
- CircleCI

AVANTAGES

- **Détecter** et **corriger** les problèmes rapidement
- Réduire les risques de **conflits de code**
- Améliorer la **qualité du code**
- Accélérer le **processus de développement**

DÉPLOIEMENT CONTINU (CD)

Le **déploiement continu (CD)** est la pratique qui consiste à **automatiser** le déploiement du code dans les environnements de production pour garantir une version **stable** et **prête à être déployée** à tout moment.

PRINCIPES

- CD facilite la livraison rapide des nouvelles fonctionnalités.
- Automatisation permet de minimiser les erreurs humaines.
- Les retours sont plus rapides avec des cycles de déploiement plus courts.

EXEMPLES D'OUTILS

- Jenkins
- Spinnaker
- Octopus Deploy
- GitLab CD

AVANTAGES

- Améliorer la rapidité de mise sur le marché
- Réduire les risques de déploiement
- Faciliter les mises à jour fréquentes
- Simplifier le processus de déploiement

EXEMPLES DE MISE EN ŒUVRE DEVOPS

SCÉNARIO : DÉPLOIEMENT AUTOMATISÉ

Le **déploiement automatisé** permet de déployer rapidement et efficacement des **applications** et des **environnements** de manière reproductible et cohérente.

PROCESSUS

1. **Développer** l'application
2. Versionner le code en utilisant **Git** ou **SVN**
3. Utiliser un service d'**intégration continue** (CI) pour automatiser les tests et la construction des artefacts
4. Utiliser un service de **déploiement continu** (CD) pour déployer automatiquement les artefacts dans un environnement de production

OUTILS UTILISÉS

- **Gestion de versions** : Git, SVN
- **CI/CD** : Jenkins, Travis CI, GitLab CI/CD, CircleCI
- **Gestion des environnements** : Docker, Kubernetes, Ansible