Section 1 DevOps : Présentation

PLAN

Qu'est ce DevOps

- Je sais que c'est la première réponse que tout le monde rechercherait et je n'essaierai pas de la definir.
- Parce que tant de gens l'ont déjà fait et devinez quoi il existe de nombreuses définitions différentes.
- Tout le monde dans l'industrie informatique en parle. L'organisation envoie des postes aux piliers pour embaucher des ingénieurs DevOps et il y a une une forte demande pour cela.
- Certains disent que c'est `automatisé`, mais d'autres disent que c'est une question de culture, ces deux croyances se contredisent.
- Eh bien, s'ils s'agit d'automatisation, les administrateurs système font de l'automatisation depuis des lustres et là, ils ont utilisé des langages de script et des outils pour y parvenir, mais ne nous ne l'appelions pas `Devops` à l'époque, nous disions simplement automatisation.
- Alors vous diriez que c'est la culture, mais si c'est juste la culture, pourquoi tant d'outils d'automisation? Vous auriez l'impression maintenant que je vous confonds, mais croyez-moi, je vais prouver certains points plus tard, continuez simplement à lire.
- Si vous lisez ceci, vous êtes en quelque sorte lié à l'industrie du logiciel. Oui, l'industriue du logiciel a pour seul objectif de créer des logiciels et de les fournir à l'utilisateur. Nous pouvons dire que depuis le tout debut, l'industrie du logiciel est divisée en deux parties:
- le `developpement et les opérations`. Le `developpement` se concentre sur la creation et le test de logiciels.
- `Les opérations` se concentrent sur la fourniture de ces logiciels à l'utilisateur sous la forme d'un site `Web` ou d'un logiciel installable. Une

fois livré, nous maintenons le logicile, nous fournissons de nouvelles fonctionnalités à nos utilisateurs et veillons à ce que le logiciel reste opérationnel et sain.

Rôles professionels et leur devise

- En developpement, nous avons des rôles distincts tels que les 'developpeurs, les testeurs de logiciels(QA), les developpeurs et architectes de base de données.
- Leur objectif est de développer toutes les fonctionnalités les plus récentes et les plus performantes du logiciel, rapidement ou rapidement.
- Dans les opérations, nous avons des rôles tels que des `administrateurs sytème, des ingénieurs cloud, des administrateurs de base de données et des professionels de la sécurité`.
- Leur Objectif est de developper toutes les fonctionnalités les plus récentes et les plus performantes du logiciel, rapidement ou rapidement.
- Dans les opérations, nous verrons des rôles tels que les administrateurs système, des ingénieurs cloud, des adminsitrateurs de base de données et des professionnels de la sécurité.
- Ici, l'objectif est de maintenir les systèmes opérationnels à tout moment. Systèmes sur lesquels est hébergé,comme vos sites Web et bases
- de données hébergés sur certains serveurs.
- Vous auriez compris maintenant que les deux parties ont des buts et des objectifs.
- L'un se concentre sur le changement rapide et l'autre sur la stabilité. Ces deux éléments sont aux antipodes:
- si nous effectuons des chnagements rapides(en ajoutant continuellement de nouvelles fonctionnalités), la stabilité devient un problème.
- Un système qui chnage continuellemnt aura des problèmes de stabilité,elle risque de perdre l'entreprise.
- Nous vivons dans un monde où nos logiciels et applications changent fréquemment. Pensez-y si vous utilisez un logiciel avec certaines anciennes foncyionnalités et qu'un autre logiciel arrive sur le marché
- avec la dernière et la meilleure fonctionnalité que vous migrerez bien sûr ver sle nouveau logiciel. Cela signifie donc que si l'équipe de developpement et d'exploitation d'une organisation ne vous offre pas
- les dernires fonctionnalités avecstabilité, elle risque de perdre l'entreprise.
- Jusquà présent, j'ai établi quelques points que je vais énumérer ci-dessous:

  - Les developpeurs visent à créer les dernières fonctionnalités rapidement et rapidement
  - L'objectif des opérations est de maintenir la stabilité des systèmes.
  - Les cangement rapides sont la demande des utilisateurs.
  - L'utilisateur a également besoin d'un logiciel ou d'applications mobiles.

- La seule et unique objectif de `DevOps` est de fournir les fonctionnalités les plus récentes et les plus performantes à l'utilisateur avec stabilité.
- Il ne s'agit pas simplement de créer de nouvelles fonctionnalités il s'agit également de fournir ces fonctionnalités à l'utilisateur, sinon à quoi bon créer
- si nous ne pouvons pas les livrer à temps. Alors, comment DevOps resout-il ce problème? Pour le comprendre, nous devons d'abord comprendre la procédure de developpement,
- puis nous nous concentrerons sur les opérations.

Processus de développemnt des logiciels

- Le `processus de developpemnt` est expliqué en détail dans les chapitres suivants, nous essaierons ici de le garder au minimum.
- L'équipe de developpment utilise un modèle de developpement logiciel pour créer le logiciel. En termes simples, ces modèles de
- developpment logiciel sont un ensemble de règles que tous les membres de l'équipe suivent pour faire avancer les choses.

1. Modèle De Cascade/Waterfall Model

- Il existe le modèle `Waterfall` qui est un modèlee traditionnel et qui ne s'applique pas bien au monde en évolution rapide d'aujourd'hui.
- Dans le modèle `Waterfall`, les phases suivantes sont suivies dans l'ordre:
  - `Exigences système et logicielles` : capturées dans un document sur les exigences du produit.
  - `Analyse`: aboutissant à des modèles, des schémas et des règles métier.
  - `Conception` : aboutissant à l'architecture logicielle.
  - `Codage` : le devloppement,la preuve et l'intégration de logiciels.
  - `Tests` : la découverte et le debogage systématique des défauts.
  - `Opérations` : l'installation, la migration, le support et la maintenance de systèmes complets.
- Ainsi le modèle `Waterfall/ En Cascade` soutient qu'il ne faut passer à une phase que lorsque la phase précédente est examinée et vérifiée.
- Ce modèle est bon pour l'équipe des opérations car elle développe simultanément l'ensemble du logiciel qu'elle peut déployer et maintenir.

- Les nouveaux changements seront également moinsfréquentes et n'imposeront pas autant de charge à l'équipe des opérations.
- Mais ce modèles ne s'adapte bien au monde actuel en évolution rapide.
- Il y a tellemnt d'inconvénients avec ce modèle et la majeure partie du developpemnt se déroule désormais avec le mdoèles `Agile`.

2. Modèle Agile

- Le `modèle agile` a developpé des logiciels en petites iétrations au lieu de developper un logiciel entier en même temps.
- La liste enitière des fonctionnalités des produits est divisé en plusieurs liste de fonctionnalités.
- Par exempl, s'il y a 50 fonctionnalités dans un logiciel, nous pouvons créer 5 listes de 10 fonctionnalités chacune.
- Désormais,les developeurs travailleront sur ces 10 fonctionnalités à la fois, créeront et fourniront ces 10 fonctionnalités lors de la première itération
- et continueront avec le reste des focntionnalités jusqu'à ce que vous obtenez le produit final.
- Chaque itération implique des équipes interfonctionnelles travaillant simultanément sur divers domaines tels:
  - Planification
  - Analyse des besoins
  - Conception
  - Codage
  - Test unitaires et
  - Test d'acceptation.
- Désormais, nous ne parlerons pas seulement de la création,mais également de sa livraison dans différents environnements tels que : `Dev, QA, Staging, UAT and Prod`.
- Déormais, cela impose une lourde charge à l'équipe des opérations, car elle doit continuellement apporter ces changements sur plusieurs environnements.
- A propos, ces différents environnements ne sont qu'un groupe de serveurs appartenant à différentes équipes, comme le contrôle de qualités appartient aux testeurs de logiciels où ils testent le logiciels.
- L'approche générale adoptée par les developpeurs est qu'une fois qu'ils ont fini de créer de nouvelles fonctionnalité, ils envoient un document procédural à l'équipe des opérations expliquant comment la déployer.
- Les developpeurs la testent sur leur machine et estime qu'elle devrait également fonctionner de la même manière en production.
- Mais les systèmes de production sont de conception différente car il y aurait `plusieurs serveurs pour le service Web, le service`

`de base de données et les services backend sécurisés par des pare-feu et NACL`.

- Il ya aurait tous les systèmes d'exploitation et logiciels de base avce une version différente decelle des systèmes des developpeurs.
- Les serveurs de developpement sont une simple machine et tous les services spnt généralement déployés sur un seul serveur.

Le problème

- Après avoir suivi le document procédural et disposnt de leurs propres compétences et connaissances,les opérations le déploieront en production.
- C'est là que le problème entre en jeu, le déploiement peut échouer, faisant échouer l'ensemble du service.
- Cela s'est produit en raison du manque de communication entre les équipes de developpement et d'exploitation.
- `Dev` ne comprend pas la partie `Ops` et l'inverse est également vrai.
- Ainsi, les opérateurs estiment que des changement fréquentscomme celui-ci peuvent casser le système et les developpeurs estiment qu'il y a trop de restrictions sur la fourniture des dernières modifications.
- Nous deveons également penser à la sécurité ici.
- Des test de sécuité sont effectués avant la mise en production. L'ensemble de decette procédure de livraison est lente et manuelle la plupart du temps.
- Pensez désormais aux changements rapides grâce au modèle agile, cela n'aide pas l'équipe des opérations à fournir le code plus rapidement.
- Ainsi, quel que soit le degré d'agilité du developpement, les opérations restent une cascade.
- Si vous réfléchissez un moment, vous comprendrez que ce n'est pas un problème technologique mais un problème culturel.
- Les deux parties de l'industrie du logiciel suivent une culture différente.
- Si cette culture ne change pas, nous ne serons pas en mesure de fournir rapidement de meilleures fonctionnalités aux utilisateurs.

Entrez Dans DevOps

- `DevOps` résoud ce problème en changeant la culture et en faisant une seule culture `Dev + Ops`.
- Il y aurait une seule équipe, `DevOps`, avec un seul objectif : une livraison rapide avec stabilité.

Mais comment?

- La première étape consiste à établir la communication et la collaboration entre `Dev et Ops`.
- Les developpeurs doivent comprendre la partie `Ops` et les `Ops` doivent comprendre la procédure de developpement.

- **`DevOps` est la pratique des ingénieurs d'exploitation et de développement participant ensemble à l'ensemble du cycle de vie du service, de la conception au support de production en passant par le processus de développement.**
- Nous avons vu précédemment dans le cycle de vie en cascade et agile que les équipes de développement et d'exploitation sont séparées, elles travaillent séparément dans leurs propres silos et ont une devise très différente.

Cycle de vie DevOps

- `Le Cycle de vie DevOps` comprend les équipes de developement et d'exploitation travaillant ensemble.
- Alors que les developpeurs travaillant sur leurs itérations agiles, les opérateurs doivent travailler à la configuration des systèmes et l'automatisation de la procédure de déploiement.
- `L'automatisation est le facteur clé ici`, car le modèle agile donne du code à plusieurs reprises pour le déployer sur les systèmes, il s'agira d'une publication continue de code
- et qui doit être déployé en continu sur de nombreux serveurs dans les environnements de developpement, d'assurance qualité, de préparation et de production.
- Si le processus de deploiement du code n'est pas automatisé,l'équipe opérationnelle doit effectuer le déploiement manuellement. Le déploiement peut inclure les procédures mentionnées ci-dessous:
  - Créez des serverus s'ils n'existent pas (cloud ou environnement virtuel)
  - Installer et configurer les prerequies ou les dépendances sur les serveurs.
  - Construisez le logiciel à partir du code source brut(si cela n'est pas fait par les developeurs).
  - Déployer des logiciels sur les serveurs.
  - Effectuez des modifications de configuration du système d'exploitation et du logiciel.
  - Surveillance de configuration.
  - Commentaires et rapport

Remarque

- Tout le processus ci-dessus peut être inférieur ou supérieur selon le type de déploiement. Nous en discuterons dans les chapitres suivants.
- Dès que nous obteneons uun nouveau changement de code, il doit être déployé sur les serveurs de production ou au moins de transfert.
- Pour cela, tout le processus doit être automatisé,nous devons d'abord automatiser le processuis de `Build` et de `Release` qui comprend:

  - Les developpeurs poussent le code dans un endroit centralisé.
  - Récupérez le code des developpeurs
  - Validez le code
  - Créer et tester le code
  - Emaballez dans un format distribuable (logiciels/artefacts).
  - Publiez-le.

- La phase suivante consiste à déployer ce logiciel publié sur des serveurs, dont nous avons déjà parlé auparavant.
- La combinaison de ce processus de construction et de publication avec le processus de déploiement nous donne le `Cycle de vie DevOps` qui est entièrement automatisé.
- Les ingénieurs `DevOps` doivent automatiser tout le processus, il doit être si transparentque lorsque les developpeurs transfèrent leur code vers un referentiel central, il doit doit être recupéré et executé via tout le processus ci-dessus et l'envoyer aux systèmes de production.

Qu'est ce que L'intégration Continue?

- Les developpeurs transmettront leur code plusieurs fois par jour vers un referentiel central.
- Chaque fois qu'il y auraun changement de code, il devra extrait, construit, testé et notifié.
- Il y aura un changement de code continu, nous devons donc continuellemnt suivre ces étapes.
- C'est pourquoi son intégration continue.
- Nous avons un chapitre séparé àce sujet où il il sera discuté en détail.
- A partir de maintenant, vous pouvez comprendre à partir du diagramme ci-dessus que de l'étape 1 à l'étape 5 est `CI`.

Qu'est ce que La Livraison Continue?

- Après `CI`, nous devrions également être en mesure de fournir nos modifications de code à tous les serveurs dans différents environnements comme `Dev, QA, Staging`.
- Il doit être automatiquement transmis aux serveurs d'assurance qualité où les testeurs effectueront des tests fonctionnel, des tests de charge, etc.
- Une fois les tests d'assurance qualité réussis, il doit automatiquement transmettre le code à la zone de préparation où le client ou un groupe d'utilisateurs peut vérifier les modifications et donner son approbation pour le déploiement à la production.

Selon Wikipédia

- `La livraison continue et DevOps` ont des significations similaires et sont souvent confondues, amis ce sont deux concepts différents.

- `DevOps` a une portée plus large et se concentre sur le `changement culturel, en particulier la collaboration des différentes équipes impliquées dans la livraison de logiciels(developpeurs, opérations, assurance, qualité, gestion, etc)`, ainsi que l'automatisation des processus de livraison de logiciles.
- La `Livraison continue` quant à elle, est une approche visant à automatiser l'aspect livraisonet se concentre sur le regroupement de différents processus et leur execution plus rapide et plus fréquente.
- Ainsi, `DevOps` peut être un produit de livraison continue et le `CD` circule directement dans `DevOps`.

Qu'est ce que le deploiement continu?

- Si l'approbation est un porcessus manuel, la `livraison de code` est une `livraison continue`, mais le processus d'approbation devient automatisé, après la mise en scène, la modification du code est effectué directement sur les systèmes de production.
- C'est ce qu'on appelle le `déploiement continu`.

DevOps et Cyce de Vie du développement logiciel

- Le cycle de vie `DevOps` ressemble à ceci:
  - Code d'enregistrement.
  - Extraire les modifications de code pour la construction.
  - Executer les tests(serveur d'intégration continue pour générer des builds et organiserles versions): tester des modèles individuels, exécuter des tests d'intégration et éxecuter des tests d'acceptation des utilisateurs.
  - Stocker les artefacts et créer un réfrentiel (dépôt pour stocker les artefacts, les résultats et les versions)
  - Déployer et publier (produit d'automatisation de publicaytion pour déployer des applications).
  - Configurer l'environnement.
  - Mettre à jour les bases de données.
  - Mettre à jour les applications.
  - Push aux utilisateurs - qui reçoivent des mises àjour d'applications testées fréquemment et sans interruption.
  - Surveillenace des performances des applications et du réseau (protection préventive)
  - Rincer et répéter.
- Le processus ci-dessus est également appelé `pipeline de livraison de code`.

Outils pour le Cycle de vie DevOps

- Nous avon,s discuté pus tôt que tout commence par la communication et la collaboration entre `DevOps et Ops`.
- Une fois que nous comprenons la culture et le processus du projet/produit, nous pouvons commencer à travailler avec tout le monde pour concevoir le `pipeline de livraison de code`.
- Nous devons décider où notre infrastructure sera hébergé? Sur le `Cloud, machines virtuelles ou machines physiques?`
- Dans le monde d'aujourd'hui, nous disposons de nombreux outils d'automatisation,mias nous devons d'abord comprendre leurs catégories.
- Quels outils sont utilisés dans quel but? Si nous ne comprenons pas, nous ne pourrons pas décider où les utiliser dans notre pipeline de livraison de code.

1. Systèmes De contôle De Version

- Est utilisé pour stocker le code source, un endroiut pour conserver tout le code et suivre sa version.
- Par exemple :
  - **Git**
  - **SVN**
  - **Mercurial**
  - **TFS**

2. Construire Des Outils

- Le processus de construction est l'endroit où nous prenons le code source brut, le testons et l'intégrons dans un logiciel.
- Ce processus est automatisé par les outils de build. Par exemple:
  - **Maven**
  - **ANT**
  - **MSBuild**
  - **Gradle**
  - **NANT**

3. outils D'intégration Continue

- Par exemple :
  - **Jenkins**
  - **Circle CI**
  - **Hudson**
  - **Bamboo**
  - **Teamcity**

4. **Outils De Gestion De Configuration**

   - Egalement connu sous le nom d'outils de configuration, ils peuvent être utilisés pour automatiser les tâches liées au système telles que `l'installation de logiciels, la configuration de service,le push/pull de fichiers, etc`.
   - Egalement utilisés pour automatiser le cloud et l'infrastructure virtuelle.
   - Par exemple :
     - **Ansible**
     - **Chef**
     - **Puppet**
     - **Slatstack**

5. **Cloud Computing**

   - Eh bien, ce n'est pas n'importe quel outil mais un `service` accessibleaux utilisateurs via internet.
   - Un service qui nous fournit des ressources de calcul pour créer des serveurs virtuels, du stockage virtuel, des réseaux, etc.
   - Il existe peu de fournisseurs sur le marché qui nous propose des services de cloud computing public. Par exemple:
     - **AWS**
     - **Azure**
     - **Google Cloud**
     - **Rackspace**

6. **Outils De Surveillance**

   - Est utilisé pour surveiller la santé de notre infrastructure et de nos applications.
   - Il nous envoie des notifications et des rapports par courrier électronique ou par d'autres moyens. Par exmeple:
     - **Nagios**
     - **Sensu**
     - **Icinga**
     - **Zenoss**
     - **Monit**

7. **Conteneurs & Microservices**

   - Cette partie sera décrit en détail dans un chapitre séparé.
   - Nous devons avoir de nombreuses connaissances en infra & Developpement pour comprendre cette catégories d'outils. Par exemple :

- **Docker**
- **RKT**
- **Kubernetes**

Cycle de vie DevOps avec des images d'outils DevOps

Résumé

- `DevOps` ne peut être défini en une seule phase. `DevOps` c'est la culture et aussi la mise en oeuvre d'outils d'automatisation. Cela dépend de quel domaine vous le définissez.
- La majorité du developpement logiciel s'effectue avec le `modèle Agile`, ce qui entraine des modfications de code de manière incrémentielle et fréquente.
- Les opérations ne s'accordent pas bien avec l'équipe Agile car les deux ont des principes différentes.
- `L'équipe Agile` veut un changement rapide, `Ops` veut maintenir la stabilité du système en n'effectuant pas de changement fréquents.
- `DevOps` aide à créer la communication, la collaboration et l'intégration entre `DSev et Ops` et au niveau de la `culture, des pratiques et des outils`.
- L'ingénieur `DevOps` doit comprendre le `cycle de vie DevOps` et mettre en oeuvre le bon outil d'automatisation au bon endroit.

L'automatisation de l'apprentissage à tous les niveaux du cycle de vie est très importante si vous souhaitez devenir un opérateur dans le domaine `DevOps`. Vous devez comprendre l'infrastructure, le developpement et l'automatisation. Plus loin dans cette partie, nous approfondirons les outils et les apprendrons. Nous comprendrons `CI&CD` du point de vue des outils.

Quizz

Question 1 :

- ***Qu'est-ce que DevOps ?***
  - *C'est une philosophie de combiner Dev & Ops au niveau de la culture, de la pratique et des outils.*

Question 2 :

- ***Qu'est ce que l'intégration continue ?***
  - *L'intégration continue est une pratique de developpement de logiciels DevOps où les developpeurs fusionnent regulièrement leurs modifications de code dans un referentiel central, après quoi des builds et des tests automatisés sont executés.*
    - https://aws.amazon.com/fr/devops/continuous-integration/

Question 3 :

- ***Qu'est ce que la livraision continue ?***
  - *La livraison continue est une pratique de developpement logiciel dans laquelle les modifications de code sont automatiquement préparés pour une mise en production.*
    - https://aws.amazon.com/fr/devops/continuous-delivery/