LOG680
Introduction à l'approche DevOps

Convertir les découvertes locales en améliorations globales

The DevOps Handbook

Part V, Chap 20

Objectifs d'apprentissage

- Expliquer comment les forums de discussion et chatbots peuvent faciliter la communication rapide au sein des équipes. Expliquer les avantages
- Expliquer pourquoi il est important de créer d'un seul dépôt de code source partagé
- Expliquer en quoi l'utilisation de tests automatisés peut permettre de diffuser les connaissances

Liste des sujets

- Introduction
- Forums de discussion et "chatbots"
- Automatisation des processus normalisés
- Création d’un seul dépôt de code source partagé
- Diffusion des connaissances en utilisant des tests automatisés
- "User stories" réutilisables
- Choix technologiques et objectifs organisationnels
- Conclusion

- **Introduction**
- Forums de discussion et "chatbots"
- Automatisation des processus normalisés
- Création d’un seul dépôt de code source partagé
- Diffusion des connaissances en utilisant des tests automatisés
- "User stories" réutilisables
- Choix technologiques et objectifs organisationnels
- Conclusion

Introduction

- Dans le chapitre précédent :
  - Nous avons discuté de la **création d'une culture d'apprentissage sécurisée** en encourageant tout le monde à parler d'erreurs et d'accidents par le biais de post-mortem sans reproche
  - Nous avons également exploré la **recherche et la correction de signaux d'échec toujours plus faibles**, ainsi que le **renforcement et la valorisation des expériences et de la prise de risques**
  - En outre, nous avons contribué à **rendre notre système de travail plus résilient** en planifiant et en testant de manière proactive les scénarios d'échec, en renforçant la sécurité de nos systèmes en détectant les défauts cachés et en les corrigeant

- Dans ce chapitre :
  - Nous allons créer des mécanismes qui permettent d'**enregistrer et de partager globalement les nouvelles connaissances** et les **améliorations découvertes localement dans l'ensemble de l'organisation**, en multipliant les effets de la connaissance et de l'amélioration globales
  - Ce faisant, nous **élevons l'état de la pratique de l'ensemble de l'organisation** afin que **chaque personne** effectuant un travail **bénéficie de l'expérience cumulative de l'organisation**

- Introduction
- **Forums de discussion et "chatbots"**
- Automatisation des processus normalisés
- Création d’un seul dépôt de code source partagé
- Diffusion des connaissances en utilisant des tests automatisés
- "User stories" réutilisables
- Choix technologiques et objectifs organisationnels
- Conclusion

Utilisation de forums de discussion et "chatbots"

- De nombreuses organisations ont créé des forums de discussion pour **faciliter la communication rapide au sein des équipes**
- Les salles de discussion **sont également utilisées pour déclencher l'automatisation**
- Technique mise au point dans le contexte d'un programme de ChatOps chez GitHub.
  - L'**objectif** était de **placer des outils d'automatisation au centre de la conversation** dans leurs salles de discussion, afin de créer une transparence et une documentation de leur travail
  - Comme le décrit Jesse Newland, ingénieur système chez GitHub

    «Même si vous êtes nouveau dans l'équipe, vous pouvez consulter le journal des discussions et voir comment tout se passe. C'est comme si vous programmiez en binôme avec eux tout le temps. »
- Ils ont créé **Hubot**
  - Application logicielle qui interagissait avec l'équipe des opérations dans leurs salles de discussion, où il pouvait être chargé d'effectuer des actions simplement en lui envoyant une commande (par exemple, «@hubot deploy owl to production»)
  - Les résultats seraient également renvoyés dans la salle de discussion

Utilisation de forums de discussion et "chatbots"

- L'**automatisation de ce travail dans une salle de discussion** (par opposition à l'exécution de scripts automatisés via une ligne de commande) présente de **nombreux avantages**, notamment:
  - Tout le monde peut voir tout ce qui se passe
  - Lors de leur premier jour de travail, les ingénieurs peuvent voir à quoi ressemble le travail quotidien et comment il est exécuté
  - Les gens sont plus enclins à demander de l'aide lorsqu'ils voient les autres s'entraider
  - Permet d'activer et accumuler un apprentissage organisationnel rapide
- Au-delà des avantages mentionnés ci-dessus, les forums de discussion **enregistrent et rendent publiques toutes les communications**
  - En revanche, les emails sont privés par défaut et les informations qu'ils contiennent ne peuvent pas être facilement découvertes ou propagées au sein d'une organisation
- **L'intégration de l'automatisation dans les forums de discussion** nous permet de **documenter et de partager nos observations et la résolution de problèmes** inhérents à notre travail
  - Cela **renforce une culture de transparence et de collaboration** dans tout ce que nous faisons

Utilisation de forums de discussion et "chatbots"

- C'est également un **moyen extrêmement efficace de convertir l'apprentissage locale en connaissances globales**
  - Chez Github, tout le personnel des opérations travaillait à distance. En fait, il n'y avait pas deux ingénieurs dans la même ville
- Github **a permis à Hubot de déclencher ses technologies d'automatisation.**
  - Notamment Puppet, Capistrano, Jenkins, resque (librairie Redis pour la création de tâches en arrière-plan) et graphme (qui génère des graphiques à partir de Graphite)
- Les **actions effectuées via Hubot** comprenaient
  - **Vérification de l'état des services**
  - **Déploiements de code en production**
  - **Désactivation des alertes** lorsque les services passaient en mode maintenance
  - **Actions effectuées à répétition**, e.g. extraction des journaux de test de fumée en cas d'échec d'un déploiement, l'exclusion de la rotation des serveurs de production, le retour en mode maître pour les services frontaux de production, ou même des excuses auprès des ingénieurs sur appel

Utilisation de forums de discussion et "chatbots"

- De même, les validations dans le référentiel de code source et les commandes qui déclenchent les déploiements en production émettent des messages sur le forum de discussion
  - À mesure que les modifications transitent dans le pipeline de déploiement, leur statut est affiché sur le forum de discussion
- Exemple d'échange sur le forum de discussion :
  - "@Sr: @jnewland, how do you get that list of big repos? disk_hogs ou something? "
  - "@Jnewland: / disk-hogs"
- Newland observe que certaines questions précédemment posées au cours d'un projet sont rarement posées maintenant. Par exemple, les ingénieurs peuvent se demander:
  - Comment se passe le déploiement?
  - Est-ce que tu déploie cette partie ou je dois le faire?
  - Comment progresse le développement du nouveau service?

Utilisation de forums de discussion et "chatbots"

- Parmi tous les **avantages** décrits par Newland, notamment l'**intégration plus rapide d'ingénieurs débutants** et l'**amélioration de la productivité de tous les ingénieurs**,

  le résultat le plus important pour lui est que **le travail d'Ops est devenu plus humain**, car les ingénieurs d'Ops ont été en mesure de découvrir les problèmes et de s'entraider rapidement et facilement

- Avec Hubot, GitHub a créé un environnement d'apprentissage local collaboratif qui pourrait être transformé en apprentissage au sein de l'organisation
- Dans le reste de ce chapitre, nous explorerons les moyens de créer et d'accélérer la diffusion de nouveaux apprentissages organisationnels

- Introduction
- Forums de discussion et "chatbots"
- Automatisation des processus normalisés
- **Création d'un seul dépôt de code source partagé**
- Diffusion des connaissances en utilisant des tests automatisés
- "User stories" réutilisables
- Choix technologiques et objectifs organisationnels
- Conclusion

Création d'un seul dépôt de code source partagé

- Un **référentiel de code source partagé à l'échelle de l'entreprise est l'un des mécanismes les plus puissants permettant d'intégrer les découvertes locales dans l'ensemble de l'organisation**
  - Lorsque nous mettons à jour un élément du référentiel de code source (une librairie partagée, par exemple), il se propage rapidement et automatiquement vers tous les services utilisant cette librairie, et il est intégré via le pipeline de déploiement de chaque équipe

- Google est l'un des plus grands exemples d'utilisation d'un référentiel de code source partagé à l'échelle de l'entreprise
  - **En 2015, Google disposait d'un seul référentiel de code source partagé contenant plus d'un milliard de fichiers et plus de deux milliards de lignes de code**
  - Ce référentiel est **utilisé par chacun des 25 000 ingénieurs et couvre toutes les propriétés de Google**, y compris les Google Search, Google Maps, Google Docs, Google+, Google Agenda, Gmail et YouTube

Création d'un seul dépôt de code source partagé

- L'un des résultats précieux de cette solution est que les ingénieurs peuvent tirer parti de l'expertise diversifiée de tous les membres de l'organisation
  - Rachel Potvin (responsable technique chez Google) supervise le groupe d'infrastructure de développeur, a déclaré à Wired que chaque ingénieur de Google pouvait accéder à «une multitude de librairies», car «presque tout a déjà été fait»

- Comme l'explique Eran Messeri (ingénieur du groupe Google Developer Infrastructure), l'un des avantages de l'utilisation d'un référentiel unique est qu'il **permet aux utilisateurs d'accéder facilement à l'intégralité du code, sous sa forme la plus récente sans besoin de coordination**

Création d'un seul dépôt de code source partagé

- Nous **devons mettre dans notre référentiel de code source partagé non seulement du code source, mais également d'autres artefacts qui codent les connaissances et l'apprentissage**, notamment:
  - Normes de configuration pour nos librairies, infrastructures et environnements (recettes Chef, manifestes de puppet, etc.)
  - Outils de déploiement
  - Normes et outils de test, y compris la sécurité
  - Outils de pipeline de déploiement
  - Outils de surveillance et d'analyse
  - Tutoriels et standards

Création d'un seul dépôt de code source partagé

- **Encoder les connaissances et les partager via ce référentiel est l'un des mécanismes les plus puissants dont nous disposons pour propager les connaissances**
- Comme Randy Shoup l'a décrit,

  « Le mécanisme le plus puissant pour prévenir les pannes chez Google est le référentiel de code unique.

  Chaque fois que quelqu'un insère quelque chose dans le référentiel, il en résulte une nouvelle construction (build), qui utilise toujours la dernière version de tout.

  Tout est construit à partir des sources plutôt que dynamiquement lié au moment de l'exécution.

  Il existe toujours une version unique d'une librairie qui est actuellement utilisée, c'est-à-dire qui est liée statiquement pendant le processus de construction. »

- Introduction
- Forums de discussion et "chatbots"
- Automatisation des processus normalisés
- Création d’un seul dépôt de code source partagé
- **Diffusion des connaissances en utilisant des tests automatisés**
- "User stories" réutilisables
- Choix technologiques et objectifs organisationnels
- Conclusion

Diffusion des connaissances en utilisant des tests automatisés

- **Lorsque des librairies partagées sont utilisées dans l'ensemble de l'organisation, nous devons permettre une propagation rapide des compétences et des améliorations**
- En s'assurant que **chacune de ces librairies comporte un nombre important de tests automatisés**, cela signifie que ces librairies s'auto-documentent et montrent aux autres ingénieurs comment les utiliser
- Cet avantage sera presque automatique si nous avons mis en place des pratiques de développement piloté par les tests (TDD), dans lesquelles des tests automatisés sont écrits avant l'écriture du code
  - **Cette discipline transforme nos suites de tests en une spécification vivante et actualisée du système**. Tout ingénieur souhaitant comprendre comment utiliser le système peut consulter la suite de tests pour trouver des exemples concrets d'utilisation de l'API du système

Diffusion des connaissances en utilisant des tests automatisés

- Dans l'idéal, **chaque librairie sera appuyée par un seul propriétaire ou une seule équipe**, fournissant la connaissance et l'expertise de la librairie
  - De plus, nous devrions (idéalement) ne permettre l'utilisation d'une seule version en production, afin de nous assurer que tout ce qui est en production exploite la meilleure connaissance collective de l'organisation
- Dans ce modèle, **le propriétaire de la librairie est également responsable de la migration** en toute sécurité de chaque groupe à l'aide du référentiel d'une version à l'autre
  - Cela nécessite à son tour une détection rapide des erreurs de régression grâce à des tests automatisés complets et à une intégration continue pour tous les systèmes utilisant la librairie

Diffusion des connaissances en utilisant des tests automatisés

- **Afin de propager plus rapidement les connaissances, nous pouvons également créer des groupes ou des forums de discussion pour chaque librairie ou service**
  - Ainsi, les personnes qui ont des questions peuvent obtenir des réponses d'autres utilisateurs, qui répondent souvent plus rapidement que les développeurs

- En utilisant ce type d'outil de communication au lieu de disposer de poches d'expertise isolées dans l'ensemble de l'organisation, nous facilitons l'échange de connaissances et d'expériences, en veillant à ce que les travailleurs puissent s'entraider pour faire face aux problèmes et aux nouveaux schémas

- Introduction
- Forums de discussion et "chatbots"
- Automatisation des processus normalisés
- Création d’un seul dépôt de code source partagé
- Diffusion des connaissances en utilisant des tests automatisés
- **"User stories" réutilisables**
- Choix technologiques et objectifs organisationnels
- Conclusion

"User stories" réutilisables

- **Lorsque des opérations ne peuvent pas être entièrement automatisées ou mises en libre-service, notre objectif est de rendre ce travail récurrent aussi reproductible et déterministe que possible**
- Nous le faisons en **standardisant le travail nécessaire**, en **automatisant autant que possible** et en **documentant notre travail de manière à permettre aux équipes produit de mieux planifier et gérer cette activité**
- Au lieu de créer manuellement des serveurs, puis de les mettre en production conformément aux listes de contrôle manuelles, nous devrions automatiser le plus possible ce travail
  - Lorsque certaines étapes ne peuvent pas être automatisées (par exemple, mettre manuellement un serveur dans un cabinet et le faire câbler par une autre équipe), nous devons définir collectivement les transferts le plus clairement possible afin de réduire les délais et les erreurs
  - Cela nous permettra également de mieux planifier ces étapes à l'avenir
  - Par exemple, nous pouvons utiliser des outils tels que Rundeck pour automatiser et exécuter des flux de travail, ou des systèmes de ticket de travail tels que JIRA ou ServiceNow

"User stories" réutilisables

- Idéalement, pour tous nos travaux récurrents d'opérations, nous devons savoir:
  - Quel travail est requis
  - Qui est requis pour l'exécuter
  - Quelles sont les étapes pour le mener à bien, etc.
- Par exemple, «nous savons qu'un déploiement à haute disponibilité se déroule en quatorze étapes et nécessite le travail de quatre équipes différentes. Les cinq dernières fois où nous l'avons effectué, il a fallu en moyenne trois jours.»
- **Tout comme nous créons des user stories dans le développement** que nous mettons dans le "backlog" puis que **nous développons, nous pouvons créer des «user stories» bien définies qui représentent des activités pouvant être réutilisées dans tous nos projets** (déploiement, capacité, sécurité, etc.)
- En créant ces user stories Ops bien définies, nous exposons le travail reproductible des opérations informatiques de manière à ce qu'il apparaisse parallèlement au travail de développement, ce qui permet une meilleure planification et des résultats plus reproductibles

- Introduction
- Forums de discussion et "chatbots"
- Automatisation des processus normalisés
- Création d’un seul dépôt de code source partagé
- Diffusion des connaissances en utilisant des tests automatisés
- "User stories" réutilisables
- **Choix technologiques et objectifs organisationnels**
- Conclusion

Choix technologiques et objectifs organisationnels

- Lorsque l'un de nos objectifs est d'optimiser la productivité des développeurs et que nous avons des architectures orientées service, les petites équipes de service peuvent potentiellement créer et exécuter leurs services dans le langage ou la structure qui répond le mieux à leurs besoins spécifiques
- Dans certains cas, c'est ce qui nous permet le mieux d'atteindre nos objectifs organisationnels
- Cependant, dans certains cas, l'inverse se produit, par exemple lorsque l'expertise d'un service critique réside dans une seule équipe et que seule cette équipe peut apporter des modifications ou résoudre des problèmes, créant ainsi un goulot d'étranglement
  - En d'autres termes, nous avons peut-être optimisé la productivité de l'équipe, mais empêché par inadvertance la réalisation des objectifs de l'organisation

Choix technologiques et objectifs organisationnels

- Cela se produit souvent lorsque nous avons un groupe Ops à vocation fonctionnelle, responsable de tout aspect du support technique
  - Dans ces scénarios, pour que nous puissions activer les compétences approfondies dans des technologies spécifiques, nous voulons nous assurer que les opérations peuvent influencer les composants utilisés en production ou leur donner la possibilité de ne pas être tenus responsables des plates-formes non prises en charge
- Si nous ne disposons pas d'une liste de technologies prises en charge par Ops, générées collectivement par Dev et Ops, nous devons systématiquement examiner l'infrastructure et les services de production, ainsi que toutes leurs dépendances actuellement prises en charge, pour rechercher celles qui créent une quantité disproportionnée d'échecs et de travail imprévu

Choix technologiques et objectifs organisationnels

- Notre **objectif** est d'**identifier les technologies qui**:
  - **Empêchent** ou **ralentissent le flux de travail**
  - **Créent** de manière disproportionnée **des niveaux élevés de travail non planifié**
  - **Créent** de manière disproportionnée **un grand nombre de demandes d'assistance**
  - **Sont les plus incompatibles avec nos résultats architecturaux souhaités** (par exemple, le débit, la stabilité, la sécurité, la fiabilité, la continuité des activités)

- **En supprimant ces infrastructures et plates-formes problématiques des technologies prises en charge par Ops, nous leur permettons de se concentrer sur l'infrastructure qui contribue le mieux à la réalisation des objectifs globaux de l'organisation**

Choix technologiques et objectifs organisationnels

- Comme le décrit Tom Limoncelli:

  «Quand j'étais chez Google, nous avions un langage compilé officiel, un langage de script officiel et un langage d'interface utilisateur officiel. Oui, d'autres langues ont été prises en charge d'une manière ou d'une autre, mais rester avec «les trois grands» signifiait prendre en charge des librairies, des outils et un moyen plus facile de trouver des collaborateurs. »

  - Ces normes ont également été renforcées par le processus de révision du code et ainsi que les langues qui étaient supportées par leurs plateformes internes

- Dans une présentation qu'il a faite avec Olivier Jacques et Rafael Garcia lors de 2015 DevOps Enterprise Summit, Ralph Loura (directeur des TI chez HP) a déclaré:

  En interne, nous avons décrit notre objectif comme étant de créer des «bouées, pas de frontières».

  Au lieu de définir des limites strictes dans lesquelles tout le monde doit rester, nous mettons en place des bouées indiquant les zones profondes du chenal où vous êtes en sécurité et soutenues.

  Vous pouvez dépasser les bouées aussi longtemps que vous respectez les principes d'organisation

  Après tout, **comment allons-nous voir la prochaine innovation qui nous aide à gagner si nous n'explorons pas et ne testons pas?**

  En tant que leaders, nous devons naviguer dans le canal, le marquer et permettre aux gens d'explorer au-delà

- Introduction
- Forums de discussion et "chatbots"
- Automatisation des processus normalisés
- Création d’un seul dépôt de code source partagé
- Diffusion des connaissances en utilisant des tests automatisés
- "User stories" réutilisables
- Choix technologiques et objectifs organisationnels
- **Conclusion**

Conclusion

- Les techniques décrites dans ce chapitre **permettent d'intégrer chaque nouvel apprentissage à la connaissance collective de l'organisation, en multipliant ses effets**
- Nous le faisons en **communiquant activement et largement de nouvelles connaissances**, par exemple via des forum de discussion (chat rooms) et des technologies telles que "architecture as code", les référentiels de code source partagés, la normalisation technologique, etc.
- En faisant cela, nous élevons l'état de la pratique des développeurs et des opérations, mais aussi de l'ensemble de l'organisation, afin que **tous ceux qui contribuent au travail le fassent avec l'expérience cumulative de l'ensemble de l'organisation**