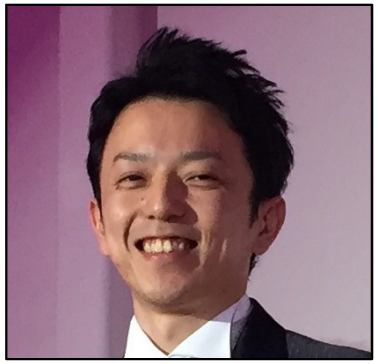
CREATIONLINE,INC.
PLANNING, PROPOSAL, CREATION, AND SOLUTIONS

DevOps導入プロセス概要

自己紹介



- 荒井 裕貴（あらいひろき）
  - クリエーションライン株式会社のChefエンジニア
  - Chef Serverの設計・導入支援
  - 講演、トレーニング
  - COOKBOOK開発(Linux / Windows)

会社紹介

CREATIONLINE,INC.

PLANNING, PROPOSAL, CREATION, AND SOLUTIONS

クリエーションライン株式会社

通称：CL (しーえる)

- プライベートクラウド基盤構築
- パブリッククラウドのコンサル
- Management Service Provider

OSS、クラウド系に強いSIer

Chefについて

- 構成管理ツールChefの開発元の会社
- 8年以上DevOpsの世界を牽引
- 日本でもDevOpsのツールといえばChefで認知
- DevOpsのインフラを支える製品が多数
  - Chef Server
  - Chef Anaytics
  - Chef Delivery
  - Chef Compliance

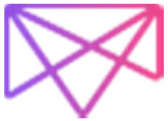
クリエーションラインはChef社のパートナー

de:code2016

de:code 2016に海外からDevOpsの黒船が襲来

Sam Guckenheimer (Microsoft Corporation)

James Casey (Chef)



Aaron Williams (Mesosphere)

Mitchell Hashimoto (Hashicorp)

目次

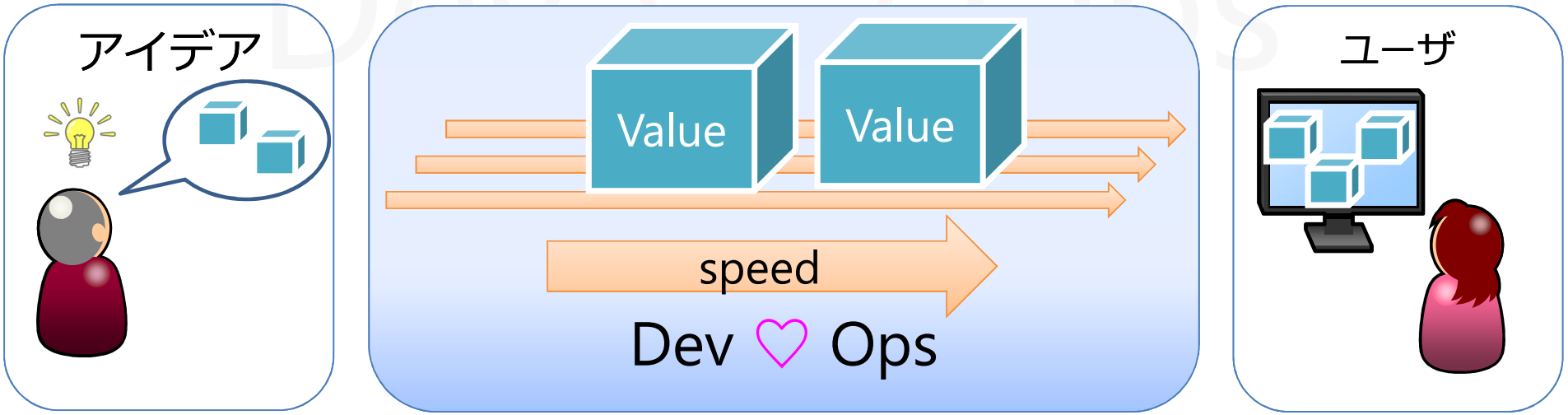
- DevOpsについて
- DevOpsの実践
- アプローチや文化の話

DevOpsとは

これがDevOpsという明確な定義は存在していない。

wikiによると・・・『DevOps（デブオプス）は、ソフトウェア開発手法の一つ。開発 (Development) と運用 (Operations) を組み合わせたかばん語であり、開発担当者と運用担当者が連携して協力する開発手法をさす。』 引用：Wikipedia (https://ja.wikipedia.org/wiki/DevOps)

⇒開発と運用がビジネスの価値提供のスピードを速めるために協力する活動



DevOpsの起源

ツール
- **自動化されたインフラ**
- **バージョン管理システムの共有**
- **ワンステップでビルドとデプロイ**
- **フィーチャーフラグ**
- **測定データの共有**
- **IRCとボット**

ツールによる
自動化/効率化
と組織の文化

文化
- **尊敬**
- **信頼**
- **障害に関する健康的な態度**
- **非難を避ける**

引用 ： 10 deploys per day (http://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr)

DevとOpsの目的の違い

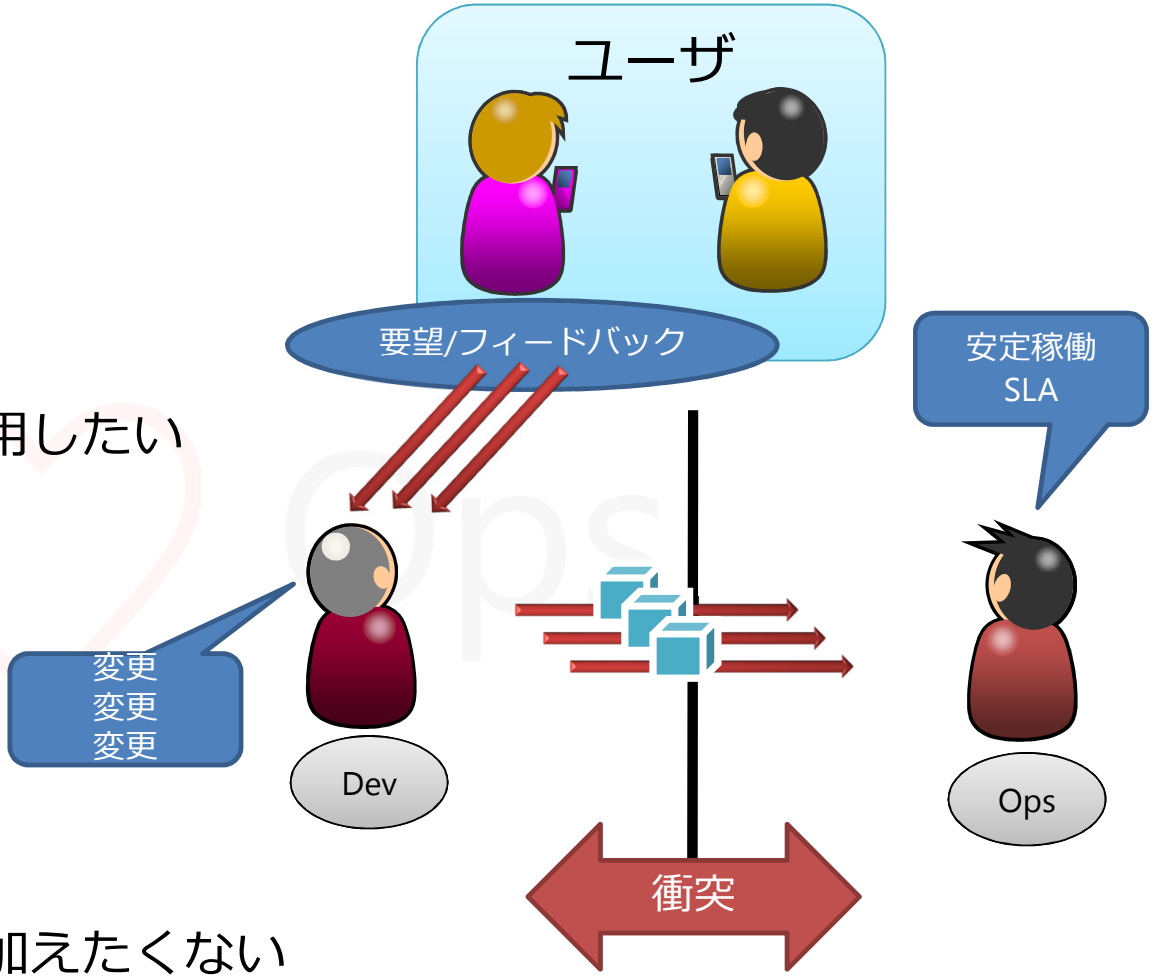
- Development
  - 新しいものを作る
  - ユーザの要望に応える

  ⇒変更は出来るだけ早く、頻繁に適用したい

- Operations
  - 安定した運用
  - SLAを守る

  ⇒スピードよりも確実性

  ⇒安定しているシステムには変更を加えたくない



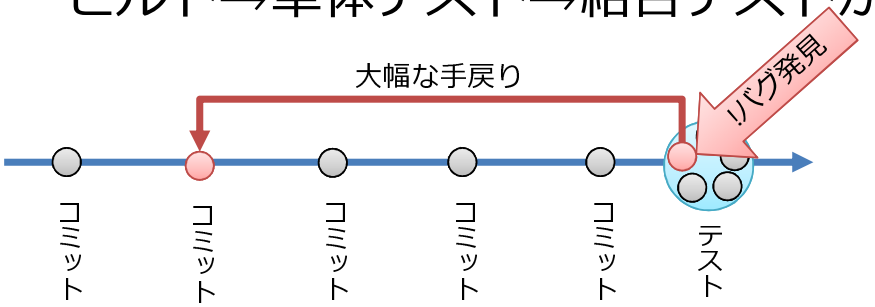
DevOpsの要素

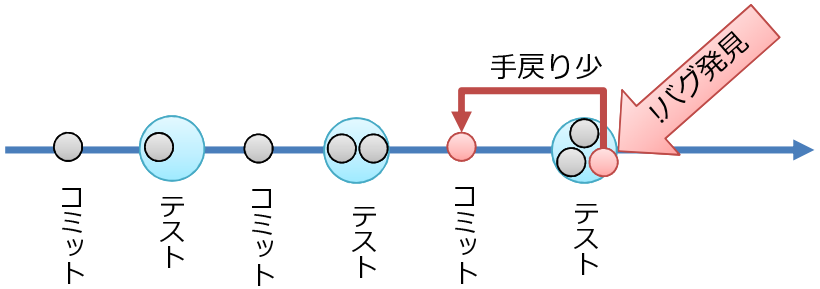
- Dev
  - アジャイル開発
  - **継続的インテグレーション**/継続的デリバリー
  - 自動テスト

- Ops
  - **Infrastructure as code**
  - 自動テスト

継続的インテグレーション：CI

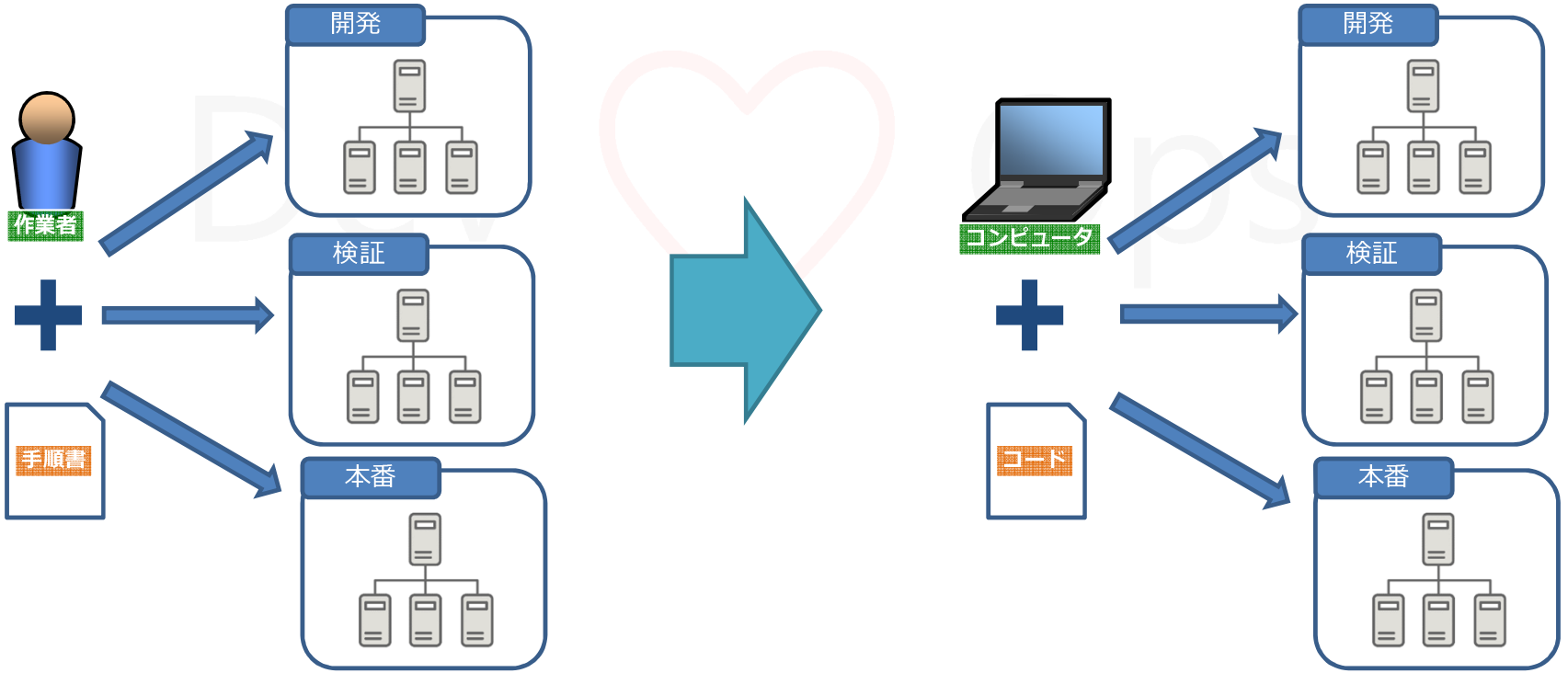
- ソフトウェアのテストを短いスパンで頻繁に行うことで問題を早期発見して、手戻りを減らしたり、ソフトウェアの品質向上を目指す取り組み
- テストを手動で頻繁に行うのは、コストがかかるが、専用のツールを利用することで作業コストを削減
- コードをコミットしたタイミングや、予めスケジュールしたタイミングで
  ビルド⇒単体テスト⇒結合テストが走って結果が通知されるような仕組みが可能





Infrastructure as Code とは(略：IaC)

- 手順をベースに手作業でやっていたインフラに関わる作業や管理を
  コードで実現できるようにしようという手法や考え方



IaCによるメリットと変化

- 冪等性 (再適用、差分の更新)
- 迅速性
- 再現性
- 正確性
- 再利用性

- アプリケーション開発の現場で培われてきた様々なワークフローをインフラの世界にも取り入れられる
  - バージョン管理システム
  - CI/CD
  - 自動テスト

従来	IaC
作業手順	コード
手順書レビュー	コードレビュー
手作業による構築	自動構築
手動によるテスト	自動テスト
Excelによる変更管理	バージョン管理システム

DevOpsの実践

DevOpsの実践

- DevOpsというものを認識する
  - ※関係者間でDevOpsの認識のずれがないようにする
- 目的を明確にする
  - マーケットにおける先行者利益を獲得するため、ソフトウェアのリリースを早める
  - 顧客からの要望に早く対応して顧客満足度を高めるために、ソフトウェアの更新サイクルを早める、など
    （ビジネスの視点が入ることが重要）（『DevOpsをやる』ではダメ）

DevOpsの実践

- 現状を分析
  - アジャイル
  - 自動化(テストやリリース)
  - インフラ(環境/自動化/IaC)
  - 承認プロセス
  - 会社ルール
  - ⇒どんなことに時間がかかっているか？
  - ⇒阻害要因はどんなものがあるか？

DevOpsの実践

- まずは一つのプロジェクトに対して適用
- 効果が出やすい部分から改善
- 承認プロセスや会社ルールの変更も検討
- 効果測定のために改善前の時間を測定

DevOpsの実践

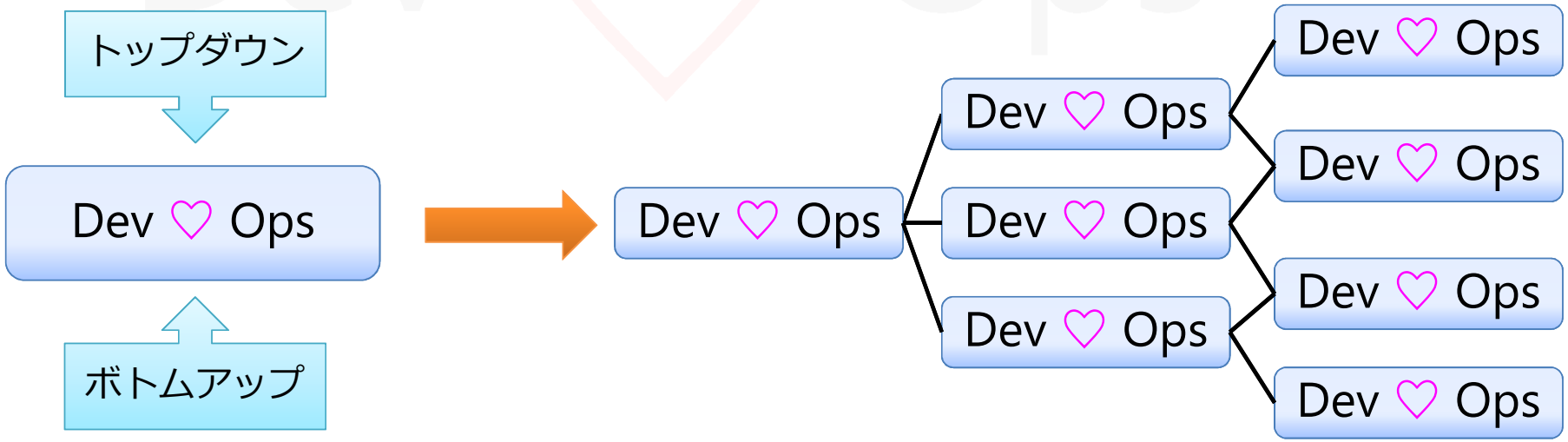
- アジャイル導入
- インフラ環境整備
- ツールの導入
  - 細かいパーツに分かれていて、組み換え可能であることが理想
- 自動化
  - 繰り返し行われる業務は自動化を検討
- 承認プロセス、フローの変更

DevOpsの実践

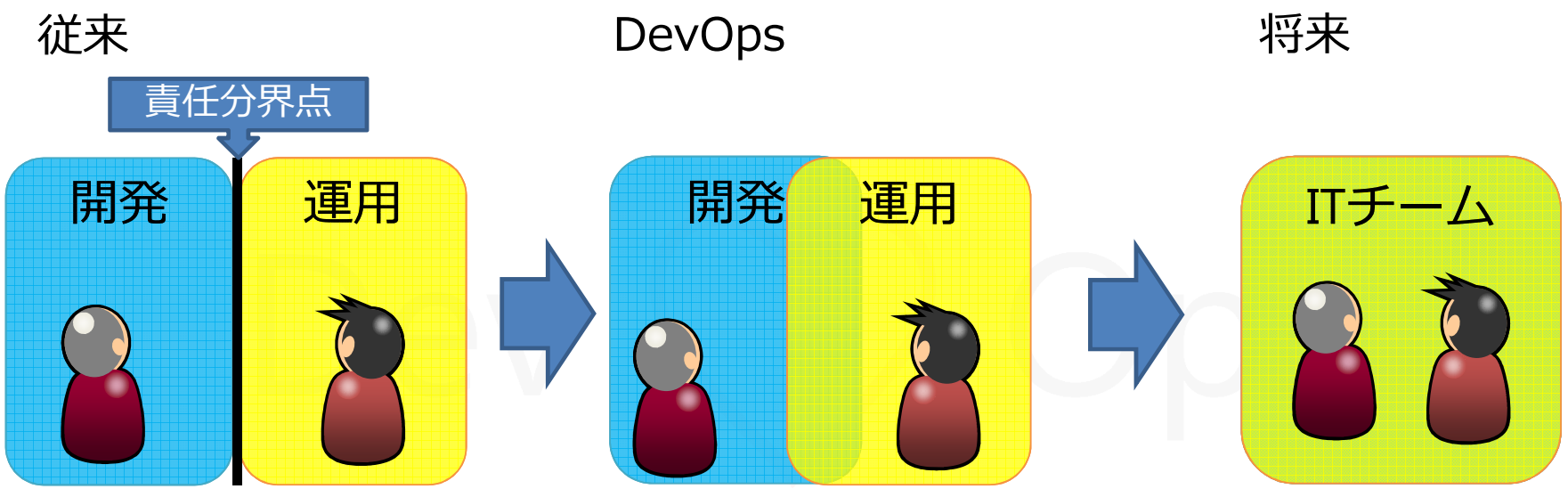
- 実施した結果を評価
- 数値で明確に表現
- 開発工数と短縮した時間の比較は慎重に
  - 開発工数と短縮した時間で評価することよりも、
    ビジネスのスピードアップにどれだけ寄与したかが重要

アプローチ

- トップダウンアプローチ
  - 経営やビジネスといった目線から全体を俯瞰
  - 承認プロセスや会社のルールに対しても視点を向けやすい
- ボトムアップアプローチ
  - 現場レベルで自動化や効率化に取り組む
  - 成果を組織内にアピールして、組織全体に活動を広げる



開発と運用の関係の変化



開発と運用の役割と責任は明確に分化

共有、競業。開発と運用の歩み寄り

明確な境界がなくなって一つのチーム

DevOpsの文化

- 尊敬
- 信頼
- 障害に関する健康的な態度
- 非難を避ける

- コラボレーション
- ムダの排除
- 変化に対しての柔軟さ
- 評価制度

DevOps関連のツール

まとめ

- DevOpsの導入は大変
  - ツールを導入すれば実現するような簡単なものではない
- DevOps = 自動化 + 文化 => ビジネスの迅速性
  - 自動化やツールの導入を進めるのに合わせて、組織の**文化**を変えていくことも大切
- 実施結果の数値化と横展開
- 実行環境の準備にCloudが便利
- まずやってみましょう