今さら聞けない人のためのDevOps超入門

日本仮想化技術株式会社
代表取締役社長兼CEO
宮原 徹(@tmiyahar)
https://VirtualTech.jp

自己紹介

- 本名 : 宮原 徹
- 1972年1月 神奈川県生まれ
- 1994年3月 中央大学法学部法律学科卒業
- 1994年4月 日本オラクル株式会社入社
  - PCサーバ向けRDBMS製品マーケティングに従事
  - Linux版Oracle8の日本市場向け出荷に貢献
- 2000年3月 株式会社デジタルデザイン 東京支社長および株式会社アクアリウムコンピューター 代表取締役社長に就任
  - 2000年6月 (株)デジタルデザイン、ナスダック・ジャパン上場(4764)
- 2001年1月 株式会社びぎねっと 設立
- 2006年12月 日本仮想化技術株式会社 設立

日本仮想化技術株式会社 概要

- 社名：日本仮想化技術株式会社
  - 英語名：VirtualTech Japan Inc.
  - 略称：日本仮想化技術／VTJ
- 設立：2006年12月
- 資本金：3,000万円
- 売上高：1億3400万円（2024年7月期）
- 本社：東京都渋谷区渋谷1-8-1
- 取締役：宮原 徹（代表取締役社長兼CEO）
- 伊藤 宏通（取締役CTO）
- スタッフ：11名（うち8名が仮想化技術専門エンジニアです）
- URL：http://VirtualTech.jp/
- 仮想化技術に関する研究および開発
  - 仮想化技術に関する各種調査
  - 仮想化技術を導入したシステムの構築・運用サポート
  - 5G活用のためのインフラ・サービス研究開発
  - DevOps支援サービスの提供
  - GPUを活用した超高速データ分析基盤「爆速DB」の提供

ベンダーニュートラルな
独立系仮想化技術の
エキスパート集団

VirtualTech Japan

「かんたんDevOps」とは

ツールとプラクティスで今すぐ始められる DevOps支援サービス

DevOpsに関する経験やノウハウがなくても、プロジェクトの最初から DevOpsで開発できるように支援いたします。

- どんなツールを使うべきか、何を行うべきかを、汎用的な**DevOps環境**と**手順書**として提供します。
- それらの使い方や、DevOpsに関する改善を行うための**サポート**を提供します。

フルマネージドでDevOpsチームに参画する 「おまかせDevOps」も提供

VirtualTech Japan

技術ブログやってます

VirtualTech Japan

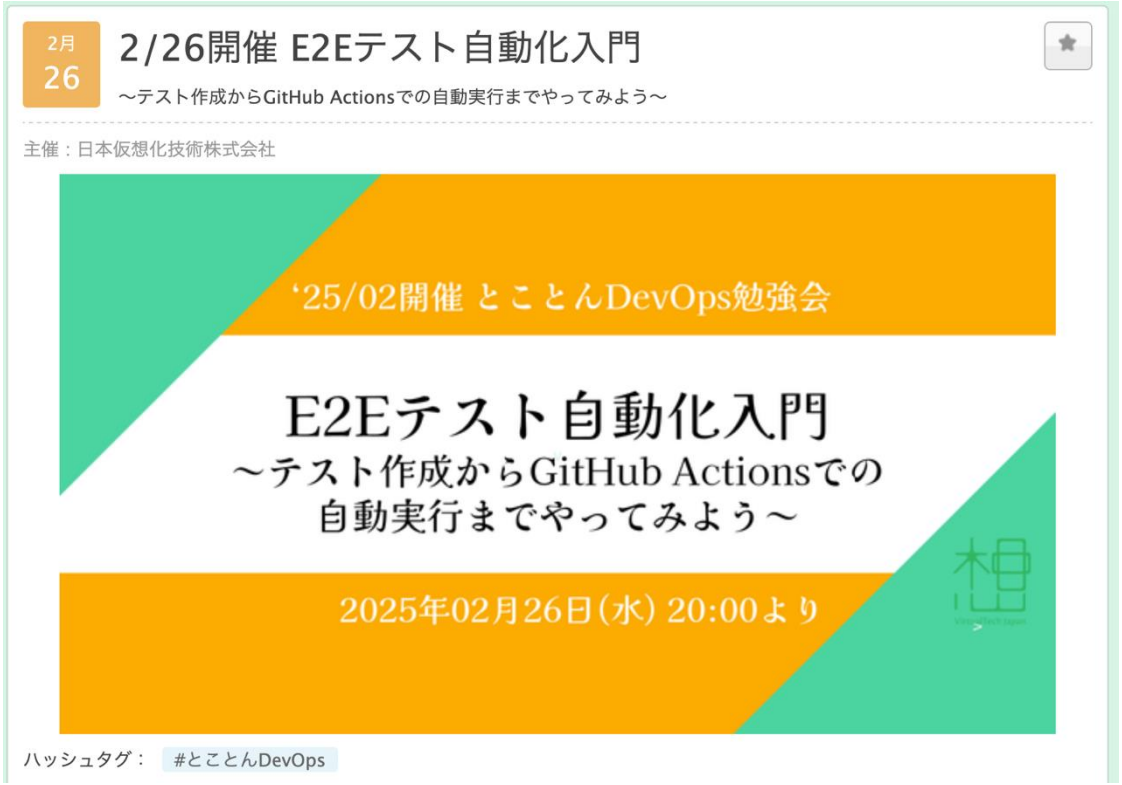
Think ITにDevOps連載あります

https://thinkit.co.jp/series/10843

VirtualTech Japan

勉強会開催してます

某弊社DevOpsチームメンバーの公開勉強会



devops-study.connpass.com

VirtualTech Japan

本日のアジェンダ

- 開発・運用の課題とDevOps
- DevOpsを支える技術
- DevOps実践のための課題と取り組み

- DXの実現やクラウドの効果的な活用のためにDevOpsの必要性が高まっている
- 自社のDevOps的な経験からエッセンスを抽出し、DevOps支援サービスを提供
- サービス開発・提供で得られた知見を解説

開発・運用の課題とDevOps

インフラ屋が如何にして
ユーザーに貢献すべきか考えた

よくある開発・運用環境の課題

- 開発と運用は分離されている
  - ウォーターフォール型
  - 縦割り（上下に横割り？）体質
- 本番リリースに時間がかかる
  - 設計からリリースまでが長大な一本道
  - 素早く繰り返すリリースを想定していない
  - バグや脆弱性が見つかっても修正されるのに時間がかかる（あるいは修正されない）

開発と運用の質的違い

- 開発者は常に変化を求める(攻め)
- 運用者は常に安定を求める(守り)

↓

- このままでは利用者のニーズに応えられず発展しない(改善されない)

↓

- 変化のリスクを組織改革とツールで低減

DevOpsとは？

- 開発と運用が密に連携して協力しあう開発手法
  - DevOpsを支援する技術やツールを積極的に利用
  - 開発は新しい機能を安心してリリースできる
  - 運用は新しい機能を安心して受け入れられる

↓

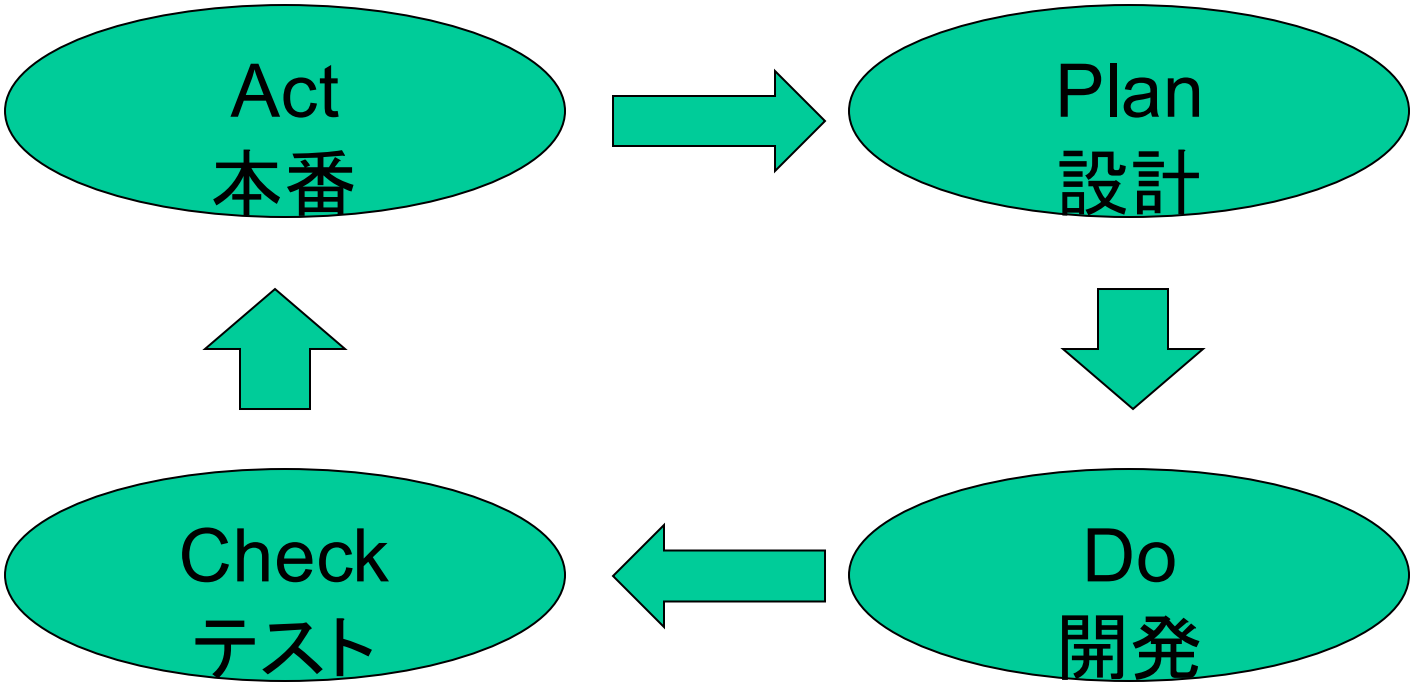
変化に対して前向きに進みやすくする

DevOpsの利点

- 開発・リリースサイクルを早く回す
  - 繰り返しリリースを前提にした開発サイクル
  - アジャイルな開発と親和性が高い
- ユーザーのニーズに迅速に応える
  - リリース期間を短縮することで要望に対し素早く応えることができる
- システムの価値が高まりビジネスへ繋げる

PDCAサイクル

DevOpsはPDCAを加速させる



DevOpsのビジネス的なメリット

マネジメント層の視点から

- 素早いリリース
- 開発状況の可視化
  - 進捗が分かる←チケット駆動開発
  - 品質が分かる←テスト駆動開発
- 開発手法の標準化
  - 開発生産性の向上

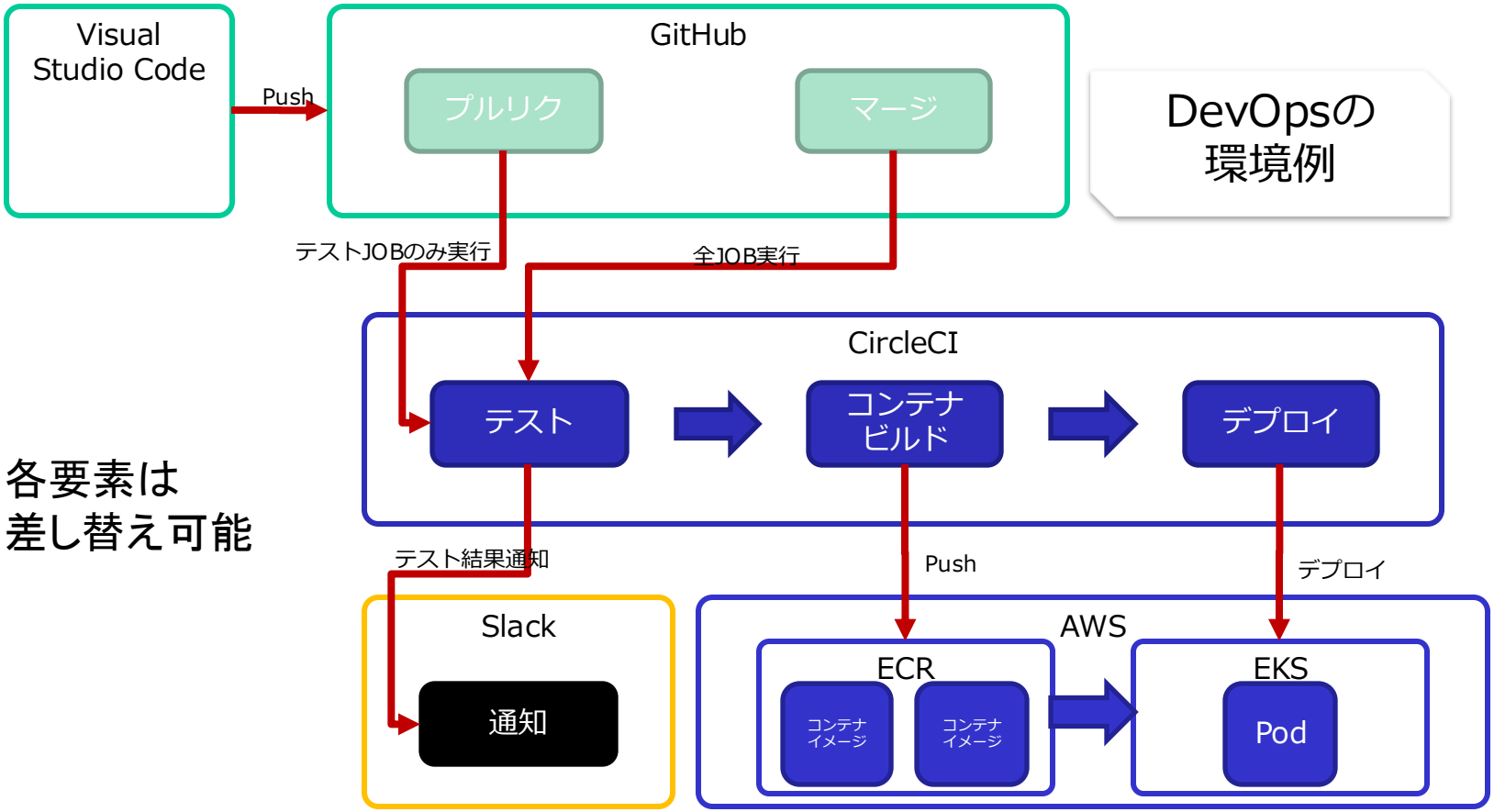
非技術的な観点からの納得感も重視

VirtualTech Japan

DevOpsを支える技術

インフラから考える
開発チームサポート

モデル環境について



各要素は
差し替え可能

DevOpsのモデルケース

- 開発環境から本番環境まで一貫した実行環境
  - コンテナベースでの実行環境で開発環境と本番環境の差異を最小化
  - アプリケーションコードのみではなく、インフラ設定もコード化し、GitHubで管理
- テストを自動化し、動かないソースコードが混入することを回避
  - プルリクエストが発行された場合、テストを自動実行。レビュー前に単体テストが確実に通る状態を保証する。
  - テストの結果はSlackに通知し、問題がある場合に即時対応可能にする
- CircleCIを使用してCI/CDのパイプラインを実行
  - ソースがマージされた場合にテストを実行した上で、GitHubで管理された設定をもとにコンテナをビルド、デプロイを実行。
  - AWS EKSを利用して、柔軟な環境構築を実現

DevOpsを支える要素・技術

- 自動化
  - テストの自動化
  - インフラ構築の自動化
- 効率化
  - 継続的インテグレーション(CI)
  - 継続的デリバリー(CD)
- チーム開発（共有化？）
  - インフラ（IaC）も含めたバージョン管理
  - チケット駆動開発によるタスクの明確化

インフラ構築の自動化（IaC）

- 手順書ベースによる構築の課題
  - 手間がかかる
  - 設定ミス等が起きやすい
- 自動化ツールによる構築
  - 仮想化、コンテナ化、クラウド化により容易に実現できるようになった
  - 誰が何度実行しても同じ結果になる
  - 設計書との相違が起きにくくなる
  - 障害復旧が容易（再構築）

継続的インテグレーション(CI)

- ビルドやテストを短期間で繰り返し行い開発効率を上げる手法
  - 定期的に実行（デイリービルド）
  - バグを早期に発見（短期間で繰り返しテスト）
  - 他のツールと連動（進捗管理等）
- 代表的なツール
  - Jenkins
  - CircleCI (SaaS)・Travis CI (SaaS)
  - GitHub Actions・GitLab CI/CD

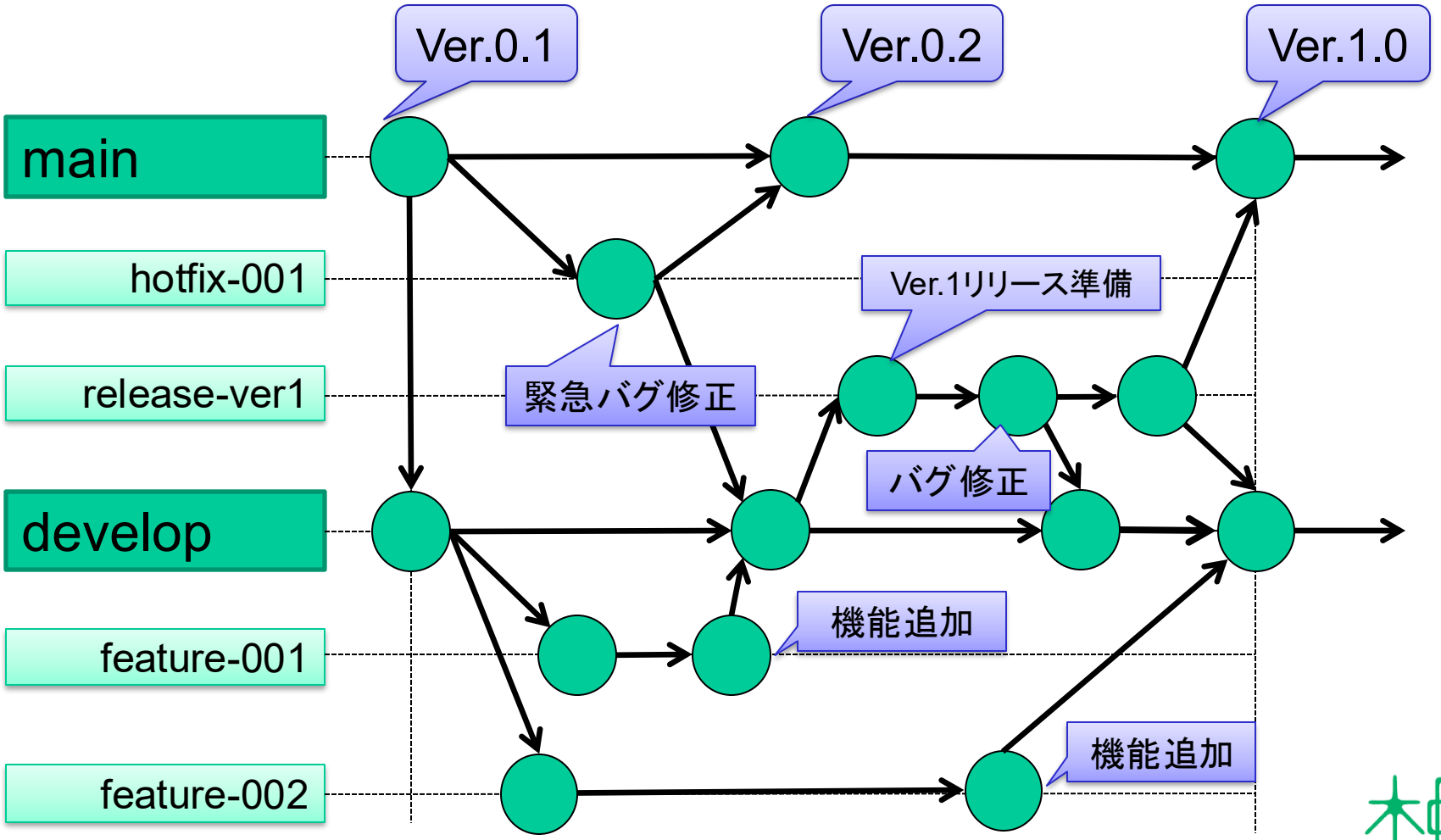
継続的デリバリー(CD)

- CIの仕組みをインフラ構築まで拡張
  - テストした成果物を自動デプロイメント
- インフラ構築自動化やコンテナと組み合わせることで短時間で構築可能
  - Kubernetesで構築されたコンテナ環境の活用
  - テスト環境と本番環境を同等の構成で構築
  - インフラを含めた結合テストを自動化

Gitを利用したバージョン管理

- ソースコード等の共有
  - 変更履歴を記録、追跡
  - 履歴の用途毎に分岐して管理(ブランチ)
  - 切り戻しが容易
- プルリクエスト機能によりレビューを可視化
- 他のツールとの連携
  - CIツール連携やGitOps
  - チケット管理ツール(Redmineなど)

git-flowの例



テスト自動化のメリット

- 人力に比べて短時間でテストが可能
- 繰り返し行うテストの省力化
- テストの品質向上
  - 一貫性の保持（属人性の排除）
  - 再現性の確保（誰が行っても再現する）
- テストツールの再利用、横展開

テスト自動化のデメリット

- 初期コストは高い
  - 継続的に利用することで費用対効果が上がる
  - 要求される仕様に大きな変更が加えられると、テストコードにも影響する
- テスト自動化は万能ではない
  - テストコードを書くのは自分たち
  - 書いたコード以上のテストはできない

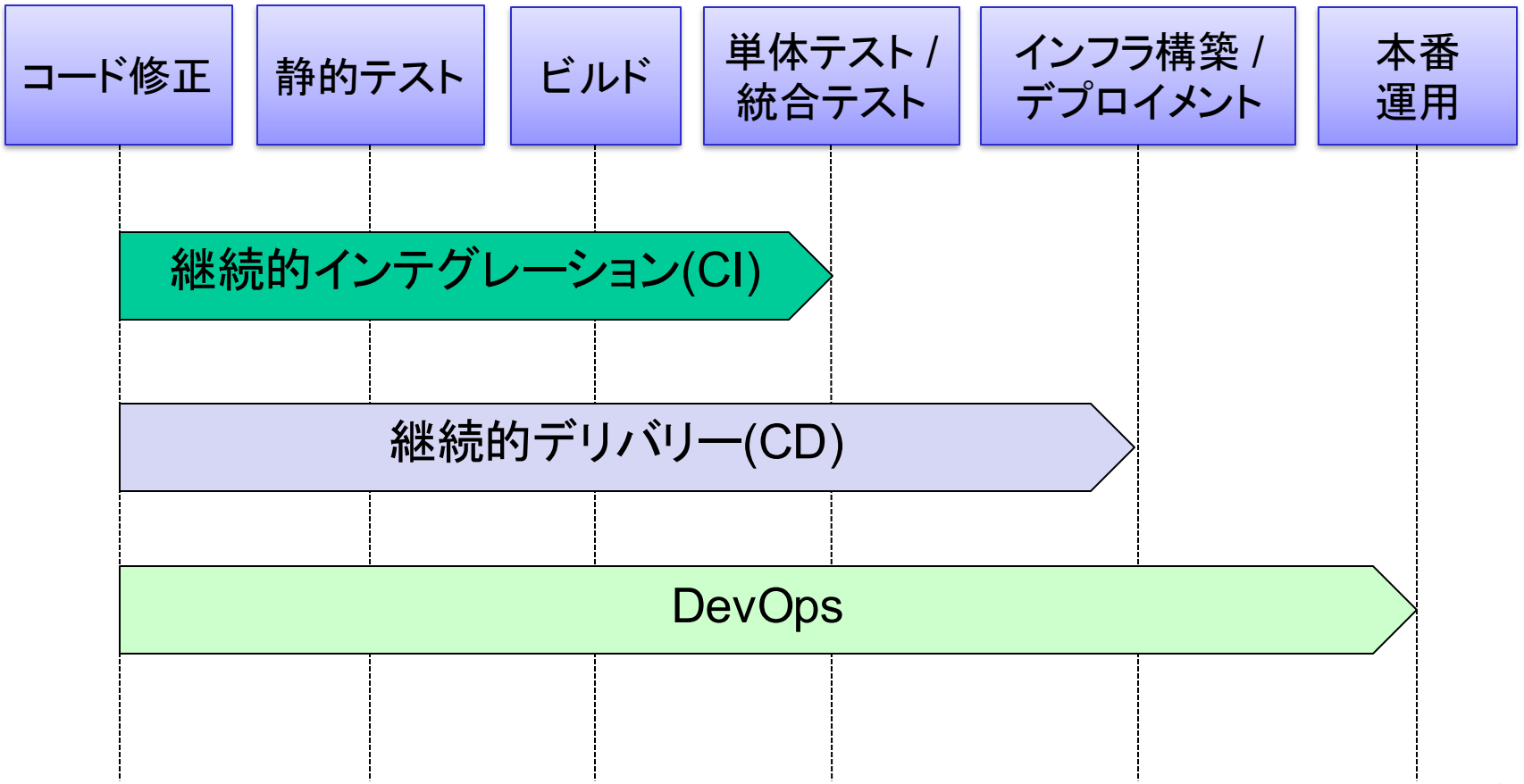
代表的なテスト

- 静的テスト
  - プログラムを実行せずコード解析のみでバグ、脆弱性、表記の揺れ等をチェック
  - レビューの自動化
- 単体テスト・ユニットテスト
  - 実装した機能（メソッド、モジュール等）単体で仕様通りに動作するかチェック
- 統合テスト・結合テスト・End to Endテスト
  - 実装した機能がうまく連携して動作するかチェック

標準化によるメリット

- プロジェクト毎に個別差がなくなる
  - 属人性が軽減される
  - 他のチームやマネージャが状況を把握しやすくなる
- フットワークが軽くなる
  - 既存のモデルやテンプレートの活用で新規案件をすばやく立ち上げられる

CI/CD/DevOpsの範囲



DevSecOps

- システム開発・運用のプロセス全体においてセキュリティを考慮する取り組み（広義）
- 開発からリリースのサイクル（CI/CD）にセキュリティ対策を織り込む（狭義）

- 要点は「テスト自動化」と「迅速なリリース」
- DevOpsを確立して改善、あるいはSecを考慮しながらDevOpsを進めていく

SBOM対応

- SBOM（Software Bill of Materials）
  - ソフトウェア部品表
  - システムを構成するソフトウェアの管理
- SPDX（The Software Package Data Exchange）、CycloneDXなどのフォーマットがある
  - SPDXはISO/IEC 5962:2021として標準化
- DevSecOps的にはコンテナ、ミドルウェアあたりをSBOM化
- SBOMはあくまで管理情報フォーマットなので脆弱性情報と連携した管理ツールが別途必要な点に注意

DevOps実践への課題と取り組み

DevOps実践のために

- 手段と目的を履き違えない
  - DevOps導入を目標にしない
  - 現状の課題をDevOps導入で解決
- 新技術を取り入れる柔軟さと協力体制
  - DevOps導入により従来とは違う「文化」を取り入れなければならない
    - 抵抗勢力も生まれやすい
  - トップダウンでスピード感のある改革が必要

CIツールの導入

- CIツールで開発の基本プロセスを自動化
- 手動で実行していることを自動化する
  - ソースコードやコンテナのビルド
  - テストの自動化
- CDの実現まで
  - インフラ構築運用の自動化

Slackを使ったChatOps

- 既存環境に影響を与えず導入可能
- メールと比較して意見を言いやすい
  - 心理的抵抗が下がる
  - チーム内のコミュニケーションが活発になる
  - LGTM (Looks Good To ME)＝いいと思うよ

Redmineを使ったチケット駆動開発

- タスク一覧とガントチャートで進捗状況を一括管理
- プロジェクト内の情報を共有、蓄積
- ChatOpsとの連動

デプロイの自動化

- エージェントレスでの導入が可能
- 学習コストが低い
- 定常作業のコード化、自動化

- 最近はTerraformも活用してます

まとめ

- 開発と運用の一体化
- 手作業から自動化への意識変革
- 安心してリリースできるサイクルの確立
- セキュリティ課題にすばやく対応
- テストへの取り組みは大きな課題
- まずはCIあたりから始めてみよう

あなたの企業に、最適な未来を。

VirtualTech Japan