# DevOpsとは？

Dev：開発（Development）

Ops：運用（Operations）

を組み合わせた言葉で、

「開発」と「運用」が連携して、よりスムーズかつ効率的にソフトウェアを提供し続けるための考え方・文化・手法を指します。

DevOpsの目的：

リリースの頻度を上げる

人的ミスを減らす（自動化）

品質を保ちつつ迅速に機能を届ける

# CI/CDとは？

CI/CDは、DevOpsの実践に欠かせない概念です。

CI（継続的インテグレーション）とは？

「コードをマージしたらすぐにテスト・ビルドして、問題がないか確認する仕組み」

チームの誰かがコードを変更したら、すぐに自動でチェック（ユニットテストやビルド）される。

目的：バグの早期発見と統合エラーの防止

CD（継続的デリバリー / デプロイメント）とは？

継続的デリバリー（Continuous Delivery）

　→ 自動で「本番環境に出す一歩手前」まで進められる状態

　（デプロイは人の判断で）

継続的デプロイメント（Continuous Deployment）

　→ テストにパスしたら自動的に本番へリリースされる

# 代表的なCI/CDツール

###### Jenkins（ジェンキンス）

老舗のCI/CD自動化ツール（オープンソース）

プラグインが豊富で柔軟性が高い

自分でサーバーにインストールして構築する必要あり

例：

# Jenkinsでできること

- Gitからソースを取得

- Mavenでビルド

- JUnitでテスト実行

- WARファイルをTomcatに自動デプロイ

GitHub Actions（ギットハブ・アクションズ）

GitHubに統合されたCI/CD機能

GitHubにPushするだけで、自動でテストやデプロイが可能

メリット：

GitHubユーザーなら簡単に導入できる

他のGitHub製品との連携がスムーズ

###### GitLab（ギットラボ）CI/CD

Gitリポジトリ＋CI/CD＋Issue管理＋Container Registryなどがオールインワン

.gitlab-ci.yml というファイルにパイプラインを定義する

GitLabサーバーを社内に構築してプライベートで使う企業も多い

###### Azure DevOps（アズール・デブオプス）

Microsoft提供のDevOpsプラットフォーム

Azure環境と親和性が高い

機能：

リポジトリ（Azure Repos）

パイプライン（Azure Pipelines）

チケット管理（Azure Boards）

例：

コードをPush → 自動テスト → Azure App Service にデプロイ

がワンストップで実行できる

# DevOps/CI/CD 導入のメリット（新人向けに）

自動化 テストやデプロイを自動で実行できる

品質向上 人手によるミスを減らし、再現性のあるプロセスを実現

迅速な開発 小さな変更でもすぐにリリースできる

チーム連携 「動くコードをすぐ共有できる」ことで無駄が減る

# まとめ（図解イメージ）

[Git Push]

↓

[CI: 自動テスト & ビルド (Jenkins / GitHub Actions など)]

↓

[CD: 自動リリース or 手動承認 (GitLab / Azure DevOps など)]

↓

[本番環境へデプロイ]

新人エンジニアがまずやるべきこと

GitHubでリポジトリを作り、GitHub Actionsを試してみる

簡単なJavaやHTMLアプリをCIでテスト・ビルドする

ツールごとの「YAMLファイルの書き方」に慣れる

Slack通知などの自動連携も体験してみる

CI/CD や DevOps は最初は少し抽象的に感じるかもしれませんが、実際に手を動かしてみると理解がグッと深まります。

by ChatGPT 2025/5/24