エンジニアのためのCI/CD再入門

# CI/CDのエキスパートが解説：CI/CDとは何か？なぜ今、必要とされるのか？

## エンジニアのためのCI/CD再入門　第1回

金 洋国（CircleCI Japan）[著]

最近、CI/CDという単語を見ることが増えてきました。Google、Microsoft、Oracleなどの大きなIT企業が自社のCI/CDツールを発表したり、CI/CDのスタートアップの買収などの話が過去一年にいくつもありました。読者のみなさまも導入はしていなくても、CI/CDについてなんとなく知っている方も多いのではないでしょうか？ 実際CI/CDに対する需要は急速に高まっています。その裏には自動化の重要性の高まりやアジャイル開発の浸透・進化があります。2回の連載でこの流れについて見ていきましょう。

### はじめに

本連載では、CI/CDを始めて学ぶ方にも、すでに知っている方にも、できるだけ新しいことを発見してもらえるように心かげました。

連載の1回目は主にCI/CDの概要と導入のメリットから始めて、CI/CDで加速する最新のアジャイル開発手法を解説します。前半は入門者向けの内容ですが、後半は一歩踏み込んだソフトウェア開発でのCI/CDの役割についてです。

連載の2回目はCI/CDサービスの代表の一つである「CircleCI」を使って実際にCI/CDの設定方法や具体的な機能について解説します。プロジェクトの登録などの導入から始めて、実際の開発現場で役立つ機能やプラクティスを紹介します。

CI/CDを最大限に活用できたとき、あなたのチームは今より数段上のソフトウェア開発が行えるようになります。この連載でそのお手伝いができれば幸いです。

### 前提条件

- CI/CDについて学びたい方
- プロダクトの品質を向上させたい方
- 最新のアジャイル開発を学びたい方
- CircleCIについて知りたい方

### 必要な環境／知識

- GitHubのアカウント
- ソフトウェアのテストについての一般的知識
- アジャイル開発についての一般知識

### 筆者について

元CircleCIの開発者で、現在はCircleCI初の海外支社であるCircleCI Japanでさまざまな活動を行っています。

## CI/CDとは

CI/CDとは「Continuous Integration／Continuous Delivery」の略で、日本語では継続的インティグレーション／継続的デリバリーといいます。CI/CDは1つの技術を指すものでなく、ソフトウェアの変更を常にテストして自動で本番環境にリリース可能な状態にしておく、ソフトウェア開発の手法を意味します。CI/CDを取り入れると、バグを素早く発見したり、変更を自動でリリースしたりできるようになります。

CI/CDには大きく分けてオンプレミス型とクラウド型があり、オンプレミス型としてはJenkins、クラウド型としてはTravis CIやCircleCIなどが有名です。オンプレミス型は一般的に拡張性が高い一方、自分たちで構築・運用する管理コストが発生します。クラウド型は拡張性が低いですが、サーバーなどを自前で用意する必要がなく、すぐに使い始めることができるのが大きな魅力です。

以下はオンプレミス型とクラウド型のメリットとデメリットの対比です。どちらを選ぶかはチーム次第ですが、よほど大規模な組織でCI/CDを専任で運用できるチームがない限り、筆者はインフラの管理コストがかからないクラウド型をおすすめします。

CI/CDサービスのオンプレミス型とクラウド型との比較

| | **オンプレミス型** | **クラウド型** |
|---|---|---|
| **メリット** | • 拡張性が高い<br>• ビルドするマシンの性能を自分たちで調整できる<br>• オープンソース（であることが多い） | • サーバーを自前で持たなくていい<br>• 簡単に始められる<br>• アップデートをしなくてよい<br>• コミュニティでナレッジを共有できる |
| **デメリット** | • 運用コストが高い<br>• サーバーを自前で用意する必要がある | • 有料（通常無料枠もあります）<br>• ビルドするマシンの性能を選べないことが多い |

## なぜ今CI/CDか?

CI/CDが近年重要視されるようになってきた理由として、2つの大きな流れがあるように思います。1つ目は自動化テストの重要性の高まり、2つ目はアジャイル開発のプラクティスの浸透と進化です。

あらゆる作業がどんどんコンピューターで行われるようになっている今日では、ソフトウェアを使ってない企業や組織はほぼないと言ってよいでしょう。これだけソフトウェアが普及していくと、当然品質に対する要求も高くなり、今まで手動でテストしていた分野でも積極的に自動化テストを取り入れるようになってきます。

顕著な例として挙げられるのが、フロントエンドのコードのテストではないでしょうか？ WebページがHTMLと少量のJavaScriptで書かれていた時代は、フロントエンドのテストを書くことは稀でした。しかし、Reactなどの高度で複雑なフレームワークが登場したり、ビジネスロジックをフロントエンドにも持たせることが多くなるにつれて、フロントエンドのテストを書くことは必須となりました。

CI/CDを使うと、コードの変更が発生するたびに自動でテストが行われるので、テストの実行忘れや、環境依存のテストを失くすことでの品質を向上させることができます。

**テストの自動化について**

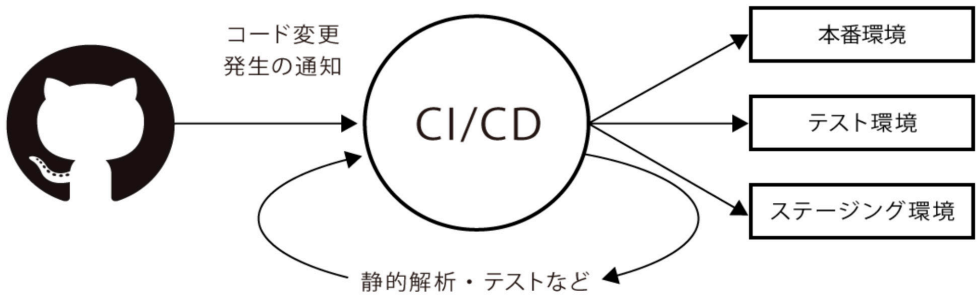
ちなみに、テストの自動化と言っても、CI/CDサービスやツールが自動でテストを作成してくれるわけではありません。あくまで、実行するテストは自分たちで書く必要があり、CI/CDは主にそれらを任意のタイミングで自動で実行してくれるだけです。

アジャイル開発の浸透と進化もCI/CDの普及を推し進めています。アジャイル開発とは小さな変更をインクリメンタルに加えていくことで、プロダクトを少しずつ開発していく手法です。この手法が実際の開発現場で有効なことが分かり、今ではより多くのチームがアジャイル開発をするようになりました。

アジャイル開発で重要なことはスピードです。より小さな粒度の変更をいかに早くテスト／リリースしてフィードバックを得るかがアジャイル開発の成功の鍵となりますが、CI/CDを使うことでこれらを実現することができます。

そして、近年アジャイル開発そのものも進化しています。もっとも目覚ましい分野として、より抽象化されたアプリケーションの運用環境の登場があると思います。DockerやKubernetesのコンテナ技術とAWSやGCPなどのクラウド技術が進歩したおかげで、本番環境にコードをデプロイすることは驚くほど簡単になりました。テストをパスした変更を常にリリースして、バグがあればロールバックを行う。こうゆう作業はもちろん手動でもできますが、CI/CDで自動化することでより効率的にできるようになります。

まとめると、CI/CDを使うとテスト自動化することで品質を高めるだけではなく、その後のリリース作業も自動化することで、よりアジャイルな開発ができるようになります。




## なぜ今CI/CDか?

読者の方にずばり質問させてください。

「テスト書いてますか?」

Yesの方はここはスキップしても大丈夫です。Noの方はもう少しだけお付き合いください。

筆者の経験上、CI/CDを活用していない場合、テストそのものを書いてないことが多い気がします。そのようなプロジェクトではテストは手動で行われます。自動化されたテストの重要性についてはすでに多くの情報があるので割愛しますが、ここではCI/CDの視点から自動化テストを始めるメリットについて解説したいと思います。

よくある誤解として、自動化するテストが少なければCI/CDを導入するメリットは少ないという考えがありますが、これはまったくの逆です。仮にテストがまだ存在しなくても、プロジェクトの初期状態からCI/CDを導入するべきです。CI/CDは建築に例えれば、基礎に分類される部分です。建物がある程度できたあとで基礎部分の変更が難しいように、プロジェクトが進んだ時点でCI/CDの導入するのは簡単ではありません。

まだCI/CD上で実行するテストがない場合は、まずはテストケースを1つ作成して同時にCI/CDの設定もしましょう。これができれば、CI/CD活用への大きな一歩を踏み出したことになります。あとは少しずつテストカバレッジを高めていけばいいのです。

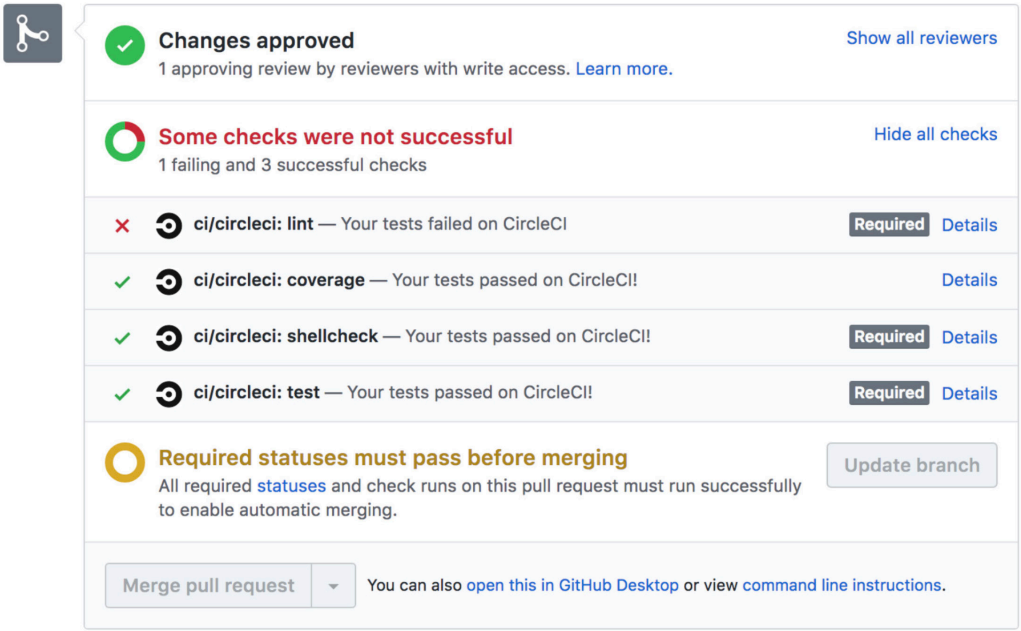
また、常にパスした状態のCI/CDが保たれるというのは気持ちがいいものです。CI/CDでテストが通るようになったら、プロジェクトのREADMEページにCI/CDのステータスバッジを追加することをおすすめします。ステータスバッジとは、現在のCI/CDの状態を表示する埋め込みコードのことです。ステータスバッジを付けることで品質の見える化が進み、チームの意識を高めることができます。以下はFacebookのReactのREADMEです。


## CI/CDは自動化ツール

コンピューターの本質がそうであるように、CI/CDの本質も自動化にあると思います。今まで手動でやっていた作業を自動化することにより、開発者はより大切な作業に時間を使うことができるようになります。ここでは簡単にCI/CDで自動化できるタスクを見てみましょう。

1つ目はテスト実行の自動化です。最近のCI/CDサービスやツールはGitHubなどのVCS（バージョン管理システム）サービスと連携していて、開発者が変更を加えるたびに、CI/CDが自動でテストを実行してくれます。また連携するCI/CD上ですべてのテストをパスしないと変更をメインのブランチにマージできないような機能もあり、これを活用すればテストが失敗したとき、その変更点を作成した開発者に修正を強制できるので、リグレッションなどを防ぎやすくなります。

以下はGitHub上で静的解析（ci/circleci: lint）のジョブが失敗しているためマージがブロックされているところです。




2つ目はリリースの自動化です。リリース作業は運用チームや開発者が手動で行うのが一般的ではないでしょうか？リリースを自動化することで工数を減らせるだけではなく、ヒューマンエラーも失くすことができます。

リリースの自動化にはいくつかのステージがあり、常に本番環境へリリース可能な状態にしておき、実際のデプロイは手動で行うこともあれば、CI/CDで一気にデプロイまでする場合もあります。どのステージがいいかは組織やプロダクトの性質ごとに異なるので一概には言えませんが、デプロイの自動化をすることで、さまざまなメリットがあります。これは大事な点なので後ほど詳しくお話しします。

3つ目にその他もろもろの作業の自動化です。コードのコンパイル、静的解析、依存関係のアップデートはもちろん、CI/CDを使うことで、さまざまなことを自動化できます。CI/CDを始めようと検討中の方に時々聞かれるのが、テストがない静的サイトのビルドができるかというご質問ですが、もちろん可能です。最近のCI/CDサービスやツールはとても柔軟に作られているので、静的サイトのビルドにかかわらず、ほとんどの作業をCI/CD上で自動化できます（ハードウェアに依存する作業などはCI/CDサービスによってはできない場合もあります）。

中には、CI/CDを使って小説の出版を行ったり、脆弱性を検知してインフラのサーバーへ自動でセキュリティアップデートをかけたりするなどの面白いの使い方をしている方ともお話したことがあります。このようにCI/CDでできることはたくさんあるので、自動化したいタスクを思いついたときは、まずCI/CDを使えないか検討してみましょう。

## CI/CDは自動化ツール

さて、ここまではCI/CDとは何か、何ができて、どんなメリットがあるかを見てきました。ここからは一歩進んで、CI/CDを活用することで、私たちのソフトウェア開発がどのように変わるかを見ていきましょう。大げさに聞こえるかもしれませんが、CI/CDはソフトウェア開発を行う組織（!）そのものまで変えてしまう大きな力を秘めています。ここからは、単なる自動化ツールを超えて、CI/CDがどのような影響をソフトウェア開発に与えるかを見ていきましょう。す。

## 継続的デプロイ

その前に一度、用語のおさらいをしましょう。まず大事なことは、継続的インティグレーション（CI）や継続的デリバリー（CD）の厳密な定義はありません。使う人や文脈によって意味が変わってきますが、一般的に継続的インティグレーションとはコードの変更を常に自動でテストすることを意味します。継続的デリバリーとは、テストをパスした変更をいつでも本番環境へリリース可能な状態にしておくことを意味します。しかし、CI/CDにはその先があります。それが「継続的デプロイ」と呼ばれるもので、テストをパスした変更を自動で本番環境へデプロイすることでリリースの自動化を行います。

継続的インティグレーションにより、ある変更がテストに通らなかった場合、その変更に問題があったことが瞬時に分かるようになります。言い換せば、テストをすべてパスした変更はリリースしても安全だということが分かります。

> **補足**
>
> もちろん現実はそんなに単純ではありません。すべてのシナリオに対してテストがあるわけではないですし、仕様バグなどはCI/CDを使っても取り除くことはできません。

こうなれば、複数の変更をまとめて一気にリリースしなくても、テストをパスした変更から順次本番環境へリリースしていくことができるようになります。

かなりレベルの高い開発チームでも継続的デプロイまで行っているチームはまだまだ少ないように思います。何年も前にAmazonは1日に最大1000回デプロイする、という話が話題になりました。この時からAmazonはCI/CDをいち早く活用して継続的デプロイをしていたようで、流石というしかありません。

**継続的デプロイをすることで以下のようなメリットがあります。**

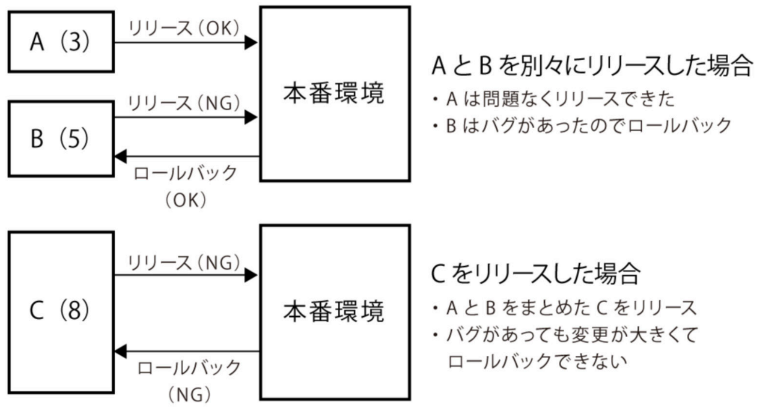
- バグの迅速な修正
- リスクマネージメント
- 変更に対するフィードバックを素早く得る

バグの迅速な修正に関しては特に説明がいらないと思うので、ここでは残りの2つについて解説していきます。

## リスクマネージメント

ソフトウェア開発に限らず、リスクマネージメントで大事なことは、発生する可能性のある問題を修正可能な範囲に抑え込むことです。つまり、問題が発生することをあらかじめ考慮して、もし発生したとしても対応可能なように計画を進めるということです。

例えば、2つの変更、AとBをリリースする場合を考えましょう。Aの変更量は3、Bは5とします。同じ内容の変更をまとめたCは8です。変更量の合計値はどちらも8ですが、別々にリリースして問題が発生した場合、AとBは個別に対応することができます。それに対し、Cは8の変更に対して一度に対応しなくてはいけません。仮に6以上の変更は複雑すぎて容易にロールバックできないとすると、最悪両方の変更に問題があったとしても、個別にリリースした場合は対応可能ですが、両方一緒にリリースした場合は対応ができません。リスクマネージメントの観点で考えると、AとBを別々にリリースしたほうが賢いと言えます。




前置きが長くなってしまいましたが、CI/CDによる継続的デプロイは前者のリスクマネージメントを可能とします。継続的デプロイができるということは、言い換えれば変更のロールバックも簡単にできるということです。問題があった変更点を取り消して再度デプロイすれば簡単にロールバックしたことになりますからね。重要なのはロールバックできる単位を小さくすることであって、デプロイのタイミングではありません。AとBを同時にデプロイしても、個別にロールバックできれば問題ありません。

この手法を使えば、別々の開発者がAとBを同じタイミングでデプロイしても、お互いが自分の修正に責任を持てれば、Cと同じビジネス価値（変更）をリリースしたことになります。同じビジネス価値なのにリスクは少ない、これがリスクマネージメントの力と言えます。

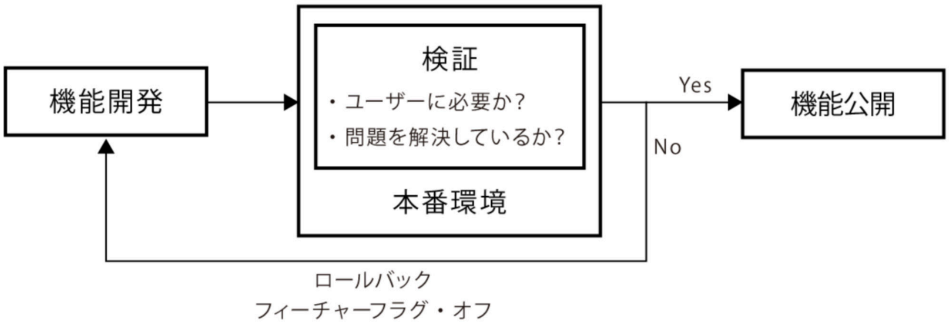
ロールバックという撤退戦略があるおかげで、開発者の仕事は劇的にやりやすくなります。仮に自分の変更に問題があったとしても、ロールバックすればいいだけなので、より創造的にコードを書くことができるようになります。もちろん、コードレビューをおろそかにしたり、見切り発車を推奨しているわけではありません。ここで言いたいことは、変更の粒度を小さくしてバグのリスクを下げることができれば、必要以上にバグを恐れる必要はないということです。筆者にとってこの”間違ってもいいんだよ”という考え方はCI/CDを通して学んだ最も大事な考え方となりました。

## フィードバックを素早く得る

スタートアップビジネスに関する名著『リーン・スタートアップ』には、いかに効率良く最低限の機能を持った試作品を作り、顧客からフィードバックを得るかがスタートアップビジネス成功の鍵だと書かれています。まったく同じことがソフトウェア開発でも言えます。つまり、MVP（実用最小限の機能）を作り、それをリリースして、フィードバックを得る。このループを素早く繰り返すことで、ユーザーが本当に求めるプロダクトを効率的に開発できるようになります。

もっと極端に言えば、多少不安要素がある機能でもどんどんリリースして本番環境でテストする、くらいの気持ちが重要だと個人的には思います。もちろん、人命が関わったりするような間違いが許されないソフトウェアもあります。しかし、そうでない分野では入念なテストよりもいかに素早く機能に対するフィードバックを得て改良するかが重要となってきます。

”フィーチャーフラグ”という言葉をご存知でしょうか？ テストしたい機能にフラグを設けて、それを外部からON／OFFにすることでその機能を見せたり、隠したりを簡単に実現する方法のことです。この方法を導入すると、ユーザーにとって価値があるかどうか分からない時でも本番環境でテストすることにより、フィードバックを素早く得ることができます。”本番環境でテスト”とはなんとも恐ろしい言葉に聞こえるかもしれませんが、10の仮説をテスト環境で検証するよりも、1度のリリースをして本番環境で検証するほうが効率的な場合は多々あります。何度もいうようにも、継続的デプロイができれば、簡単にロールバックできます。ロールバックの速さではフィーチャーフラグには一歩譲りますが、継続デプロイを使うことで本番環境からより早くフィードバックを得ることができるようになるのです。




素早いフィードバックループについてはそれだけで一冊の本が書けてしまうくらい重要なアイデアです。プロダクト開発だけではなく、組織の構成そのものも変えてしまうほどの力がありますが、話が抽象的になってしまうのでここでは取り扱いません。興味がある方は「Continuous Development Will Change Organizations as Much as Agile Did」などの記事を読んでみることをおすすめいたします。

## まとめ

なぜ今CI/CDが注目されるのかが、分かっていただけたでしょうか？ CI/CDを使うことでさまざまな作業を自動化できるだけではなく、デプロイとロールバックを自動化することでより効率的なプロダクト開発ができるようにないます。

さあ抽象的なCI/CDの話もそろそろ終わりにしましょう。読者の多くの方は、実際にCI/CDを設定するのか知りたくてウズウズされているころかもしれません。次回はクラウド型の代表サービスの一つである「CircleCI」を使って、もっと具体的にCI/CDの機能を見ていくのでご期待ください。

**著者プロフィール**

**金 洋国（CircleCI Japan）（キム ヒロクニ）**

CircleCIで2.0などのプロダクト開発に携わった後、CircleCI Japanを立ち上げてからはTech Leadとして技術全般を担当。趣味は電動キックボードで日本で普及するように様々な活動をしています。

※プロフィールは、執筆時点、または直近の記事の寄稿時点での内容です