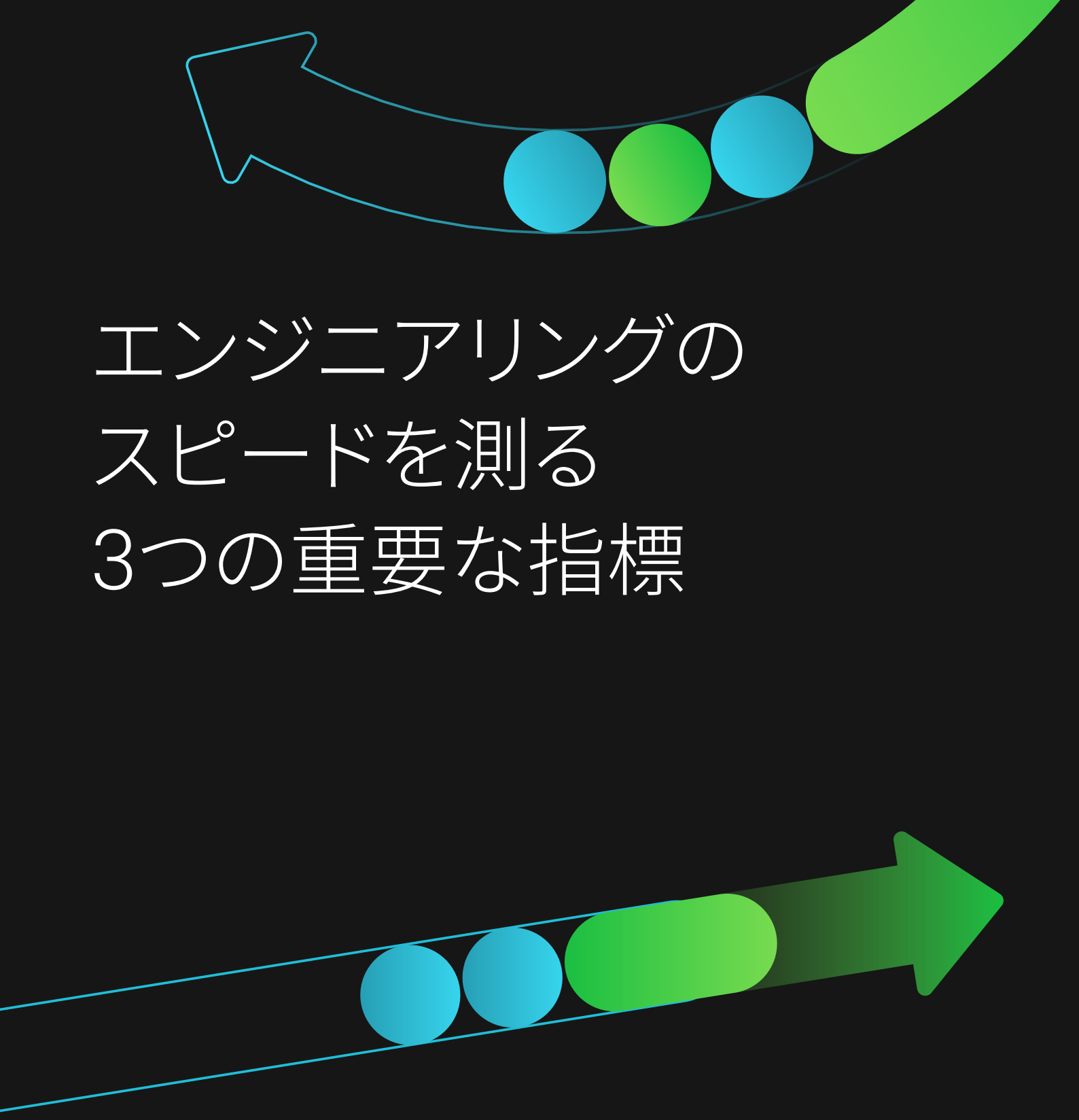


エンジニアリングのスピードを測る3つの重要な指標

circleci

目次

3ページ

はじめに

目的
調査手法

7ページ

3つの指標

メインラインブランチの安定性
デプロイ時間
デプロイ頻度

11ページ

調査結果、分析、ベストプラクティス

メインラインブランチの安定性
デプロイ時間
デプロイ頻度
指標の相互関係

22ページ

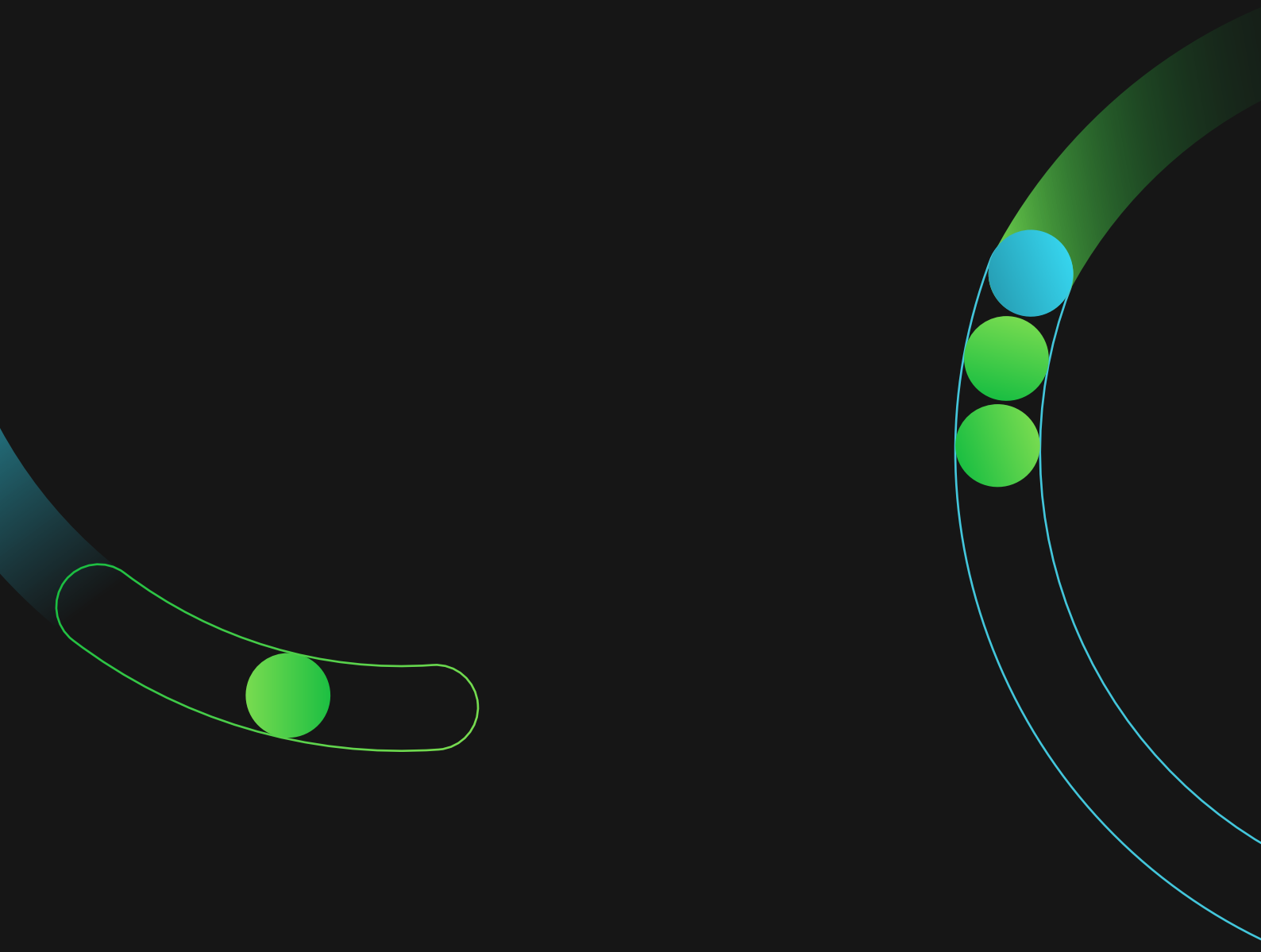
デモグラフィック

地理的分布
ビルドを実行するユーザー数
主要なプログラミング言語

25ページ

まとめ

はじめに



エンジニアリングリーダーはもちろん、ソフトウェアを柱としてビジネスを展開している企業の経営陣なら、おそらく多くの時間を費やして DevOps について検討していることでしょう。

DevOps とは、ソフトウェアの世界では一般的な「開発」と「運用」という 2つの部門が連携して協力する手法です。この 2つのチーム間の壁をなくすことで、企業は安定性やパフォーマンスを確保しながら価値ある製品を迅速に開発できます。ただし、DevOps という 1つのチームが存在するわけではありません。DevOps とは文化的なムーブメントであり、今やコード配信のスタンダードとなりつつあります。

業界を問わず、DevOps の手法を採用している企業は成功に近いと言われ、エンジニアリングリーダーの大半が DevOps のアイデアを支持していますが、DevOps の手法を導入しようとしても、その段階でつまずいてしまうことが少なくありません。

導入障壁としては、組織内の軋轢、煩雑な変更管理、惰性的な文化の蔓延など、多数の要因が挙げられるでしょう。しかし突き詰めてみると、理解度の低さがその根底にあるようです。測定できないものを管理できるはずがありません。追跡すべき指標や進捗状況を定義する方法がわからなければ、デジタルトランスフォーメーションが正しい方向に進んでいるかどうかもわかりません。

目的

CircleCI は、ソフトウェア開発の重要な領域を担う CI/CD プラットフォームを提供しており、このプラットフォームでは、メインラインブランチの安定性 (デプロイ可能な状態にある割合)、コミットからデプロイまでの時間 (CDT)、デプロイの頻度を把握できます。そして、これらの 3つの指標から組織のスピードと成長を的確に予測できると考えます。

本レポートの目標は、**1) これらの指標とビジネス成長の相関関係を探ること、2) これらの指標を改善するために実践できる具体的なベストプラクティスを明らかにすること**です。ソフトウェア開発ライフサイクルに深くかかわる CircleCI だからこそ得られる貴重なデータを分析したところ、CircleCI ユーザーの皆様は、わずか 1時間のうちに約 5,400 のブランチをビルドしています。これらのビルド時間を合計すると、約 3,900時間にのぼります。本レポートの執筆時点で、CircleCI は月間 620万以上のビルドを処理しています[1]。

こうしたデータを基に、指標ごとに特に優れたパフォーマンスを上げているチームを特定したうえで、各チームにソフトウェアの開発手法やデプロイ手法についてインタビューしました。その中から明らかになったベストプラクティスを紹介しながら、皆様が DevOps の原則を導入するための具体的な方策をお伝えしたいと思います。

---

[1] これらのビルドを支えるインフラストラクチャの詳細については、StackShare の記事 (英語) をご覧ください。

調査手法

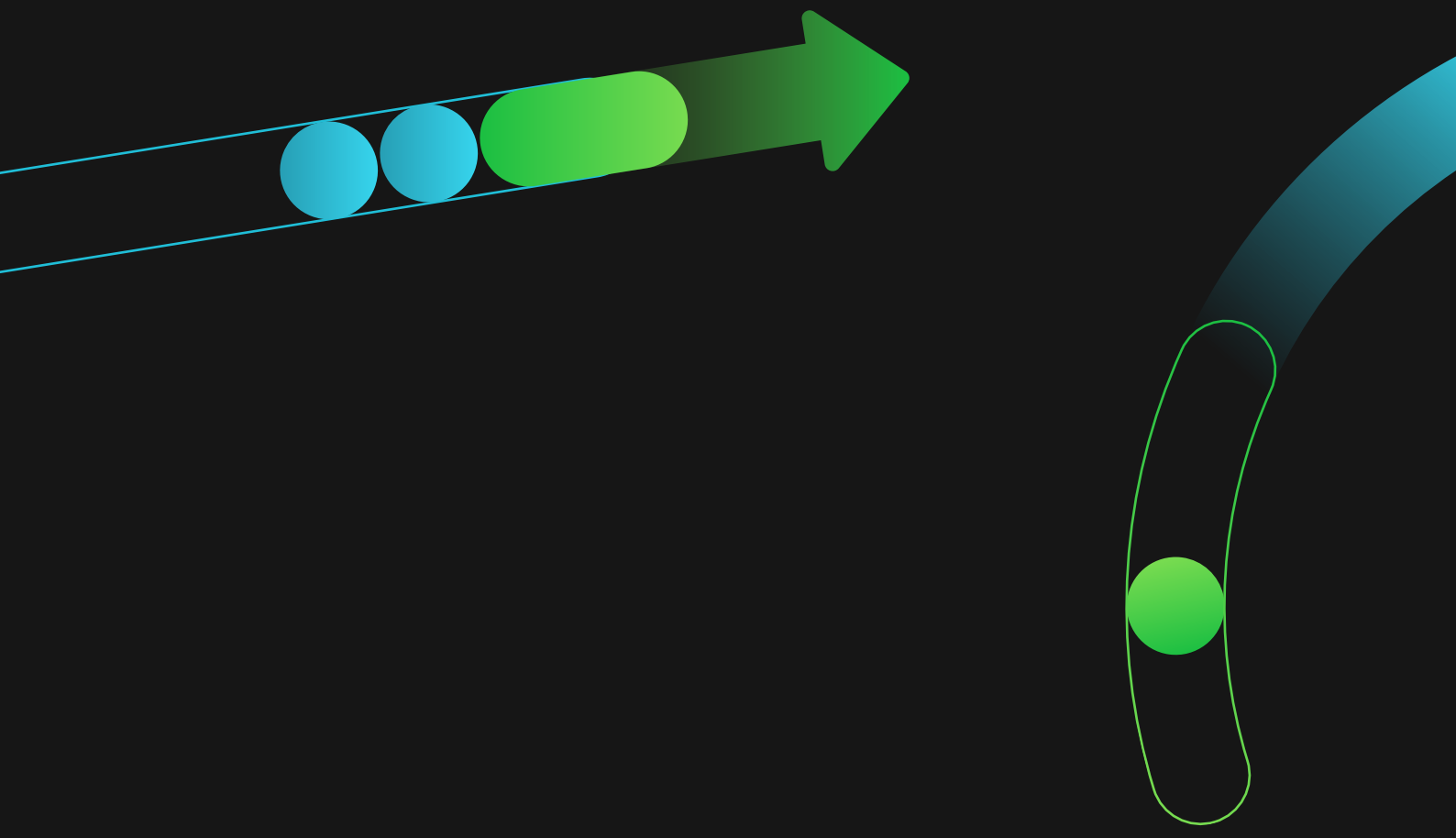
本レポートでは、2017年 6月 1日から 7月 31日までに CircleCI のクラウドプラットフォーム上で GitHub または Bitbucket を利用してビルドを行った組織をサンプルとして調査しました[2]。稼働していないプロジェクトの影響を排除するため、a) 過去にプロジェクトのテストとデプロイを行ったことがあり、b) 期間中に 1つ以上のプロジェクトで 1回は master へのマージを行った組織を対象としました。次回の調査では、クラスタリングや分類などによって、あまり条件を設けないようにする予定です。

成長の指標として Alexa Internet の世界ランキングを採用し、Clearbit (英語) のエンリッチメントサービスを使用して属性を分析しました。組織と Clearbit のエンティティー致率は 82.02 % でした。次回の調査では、他のソースを使用して同等のドメインを得られるようにする予定です。

---

[2] CircleCI はセキュリティとプライバシー保護に真摯に取り組んでおり、本レポートで紹介するお客様からは掲載許可を頂いています。詳細については、当社のプライバシーポリシー (英語) をご覧ください。

3つの指標



メインラインブランチの安定性

「今すぐデプロイする必要が生じたとして、デプロイできますか?」

メインラインブランチは、アプリケーションの信頼できる唯一のソースであり、開発者が各機能ブランチを作成するときの原型となります。メインラインブランチが壊れると、開発作業は停滞します。新機能のビルドを開始することも、重大なインシデントに対応することもできません。

メインラインブランチの安定性とは、どれだけ**デプロイ可能な状態**であるかを示し、「今すぐデプロイする必要が生じたとして、デプロイできますか?」という問いに対する答えとなります。その答えが「いいえ」なら、デプロイにかかる時間は重要ではありません。デプロイ自体が不可能だからです。つまり、メインラインブランチの安定性はデプロイ頻度にも直結します。

今回の調査では、プロジェクトのデフォルトブランチが失敗状態であった実測時間の割合を「安定性」と定義しました。ここで言う「実測時間」とは、実際に経過した時間を指します。これは、プロジェクトのすべてのコンテナ (ソフトウェアのテストを行う仮想マシン) によって消費される合計時間とは異なります。デフォルトブランチとは、プロジェクトの「master ブランチ」として指定しているブランチを指します。

デプロイ時間

デプロイ時間が短いほど、製品の変更にかかるコストは抑えられます。

コードの作成、レビュー、テストを終えても、その後さらにユーザーにデリバリーしなければなりません。コードをメインラインブランチから本番環境へと移す作業は、数分で済む場合もあれば、数時間かかることもあります。ブランチの目的が新機能でもバグ修正でも、このコストは組織のコードベースが変更されるたびに発生します。

デプロイ時間からは、デプロイのコストを評価できます。デプロイ時間が短いほど、製品の変更にかかるコストは抑えられます。エンジニアにとってはデプロイを待つ無駄な時間が減り、次の作業にすばやく取り掛かれるようになります。プロダクトオーナーはより多くのテストを行い、より多くのプロトタイプを作成できます。その結果、ユーザーがアップデートを手にするまでの時間が短縮され、バグの発見から数分でパッチが適用されます。

今回の調査では、ビルドのキューイングからビルドの完了までの実測時間 (分単位) をデプロイ時間と定義しました。

デプロイ頻度

> 組織の「心拍」を示すバイタルサインです。1回鼓動を打つたびに、
> 価値を提供し、顧客のニーズを発見し、問題を修正しているのです。

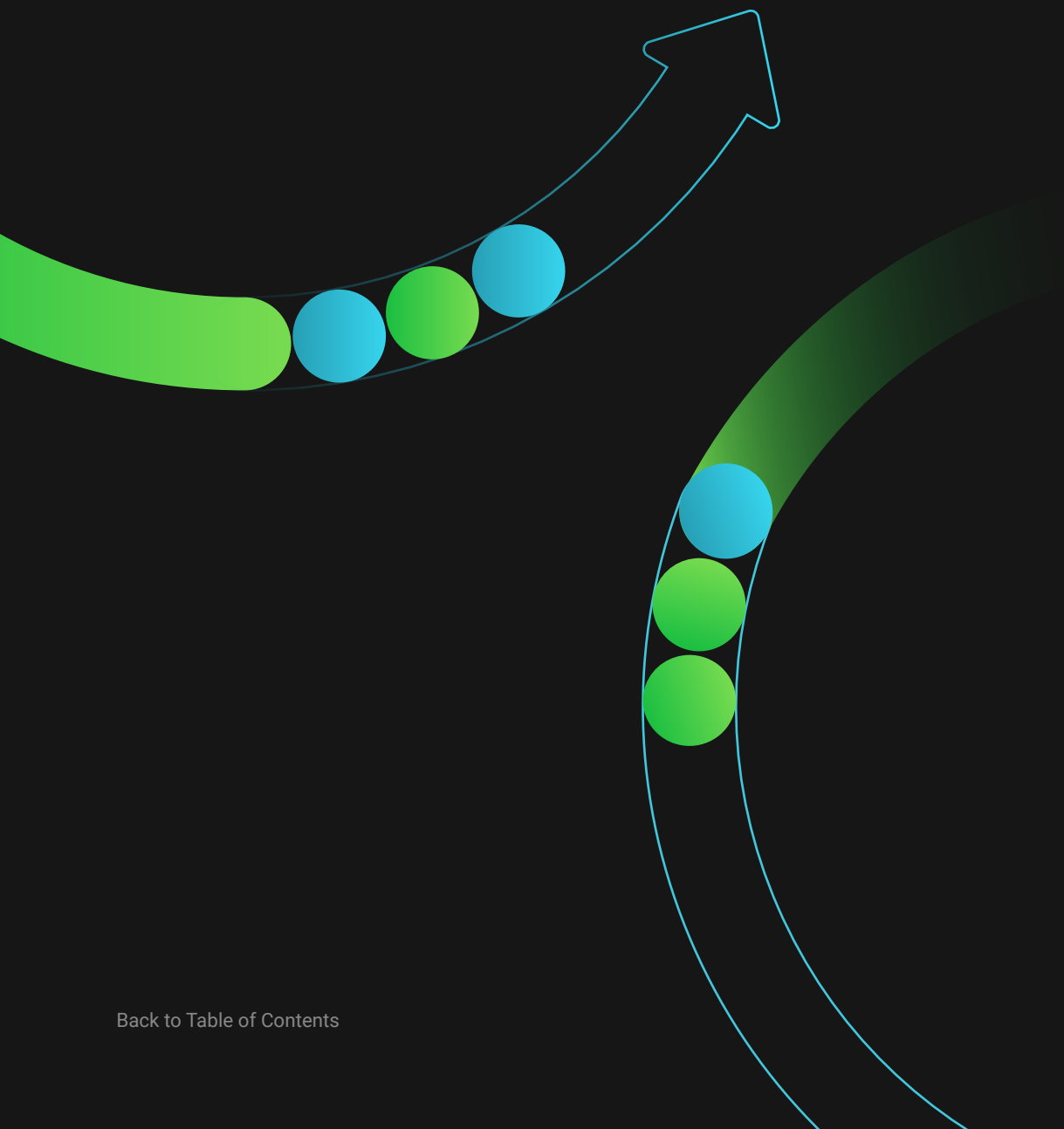
前述の指標は、どれくらいの間隔で変更を配信できるかに関するベンチマークですが、組織の実際のスピードを示すのはこのデプロイ頻度です。前述の 2つの指標と直接関係しており、安定性の向上とデプロイ時間の短縮は、どちらもデプロイ頻度の向上につながります。

デプロイ頻度は組織の「心拍」を示すバイタルサインです。1回鼓動を打つたびに、価値を提供し、顧客のニーズを発見し、問題を修正しているのです。

『Accelerate』の著者であり、DevOps Research and Assessment (DORA) の CEO 兼チーフサイエンティストを務める Nicole Forsgren 博士は、次のように述べています。「高パフォーマンスの組織は必要なときにすぐデプロイできますが、低パフォーマンスの組織がコードをデプロイできる頻度はせいぜい月に 1回です。このスピードの差は、価値を提供し、顧客の満足度を向上させ、コンプライアンスや規制の変更に対応する能力に直結します」

今回の調査では、有効なデプロイ手順に沿って、組織単位で 1週間のうちに CircleCI のクラウドプラットフォーム上でデフォルトブランチのビルドを実行した回数の中央値をデプロイ頻度と定義しました。

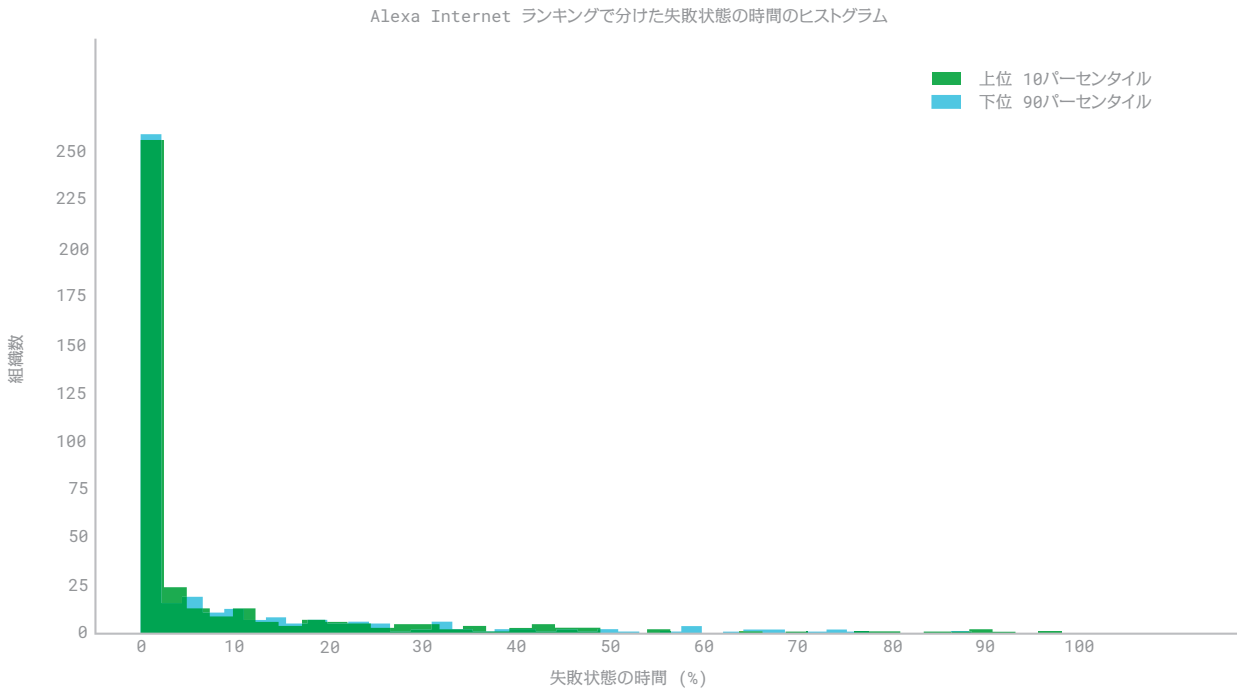
調査結果、分析、ベストプラクティス



メインラインブランチの安定性

全体的に、安定性は組織にとって重要な指標であり、80 % の組織が 90 % 以上の時間をメインラインブランチがデプロイ可能な状態に保っています。データは指数分布となり、95パーセンタイルに位置する組織は 99.9 % の時間、安定性を維持しています。安定性の中央値は 98.7 % である一方、下位 5パーセンタイルの組織では 47 % の時間をメインラインブランチが失敗状態のままにしています。

Alexa Internet ランキングの上位 10パーセンタイルの組織についても同様の分布が見られ、下位 5パーセンタイルの組織では 46 % の時間が失敗状態にありました。安定性の中央値は 98.5 % で、上位 5パーセンタイルの組織では 99.9 % です。全体の結果と同様に、80 % の組織が 90 % の時間をメインラインブランチが安定した状態に保っています。



Fortune 100 企業でも少人数のスタートアップ企業でも、大半の組織はメインラインブランチの安定性の価値を理解しているようです。しかし、高い安定性を実現して維持するためには、チームの構造を見直したり、従来のプロセスを変更したりすることが必要になります。

ベストプラクティス

堅牢なテスト

メインラインブランチが失敗状態になると、開発作業は停止します。優れた企業はこのことを理解しているため、堅牢な自動テストに投資しています。変更ごとにテストすることで、エンジニアはコードが想定どおりに機能することを確認できます。こうした一連のテストはソフトウェアの「予防治療」として作用し、定期的に行うことではるかに簡単にバグを検出できるようになります。頻度が高ければさらに効果的です。

Code for America (英語) のシニアソフトウェアエンジニアを務める Ben Sheldon 氏は、適切なテストを実施することで作業者の認知負荷を軽減できると考えています。「私たちは、テストに合格すれば、実際に不具合が発生するリスクは低くなると強く確信しています。そのため、コードレビュープロセスでは、問題を防止することよりも、変更内容について理解することを重視しています。究極的には、アーキテクチャやコードベースの長期的な方向性について、エンジニアが高いレベルで議論を交わせるようにしたいと考えています」

テストを行えば、メインラインブランチからバグを排除できるだけでなく、エンジニアの貴重な精神的エネルギーを重要な意思決定に費やすことができます。些細な判断について議論して時間を浪費する代わりに、システムの選択肢について高い次元で吟味できるようになれば、全体的なコード品質が向上し、ひいては安定性が向上します。

フィーチャーフラグ

非常に高度なテストでも、実際のアプリケーションの複雑さを捉えることはできません。CircleCI では、テストによって潜在的なあらゆる問題を検出する代わりに、機能のリリースに LaunchDarkly のフィーチャーフラグを利用して、リリースの安全性を高めています。

LaunchDarkly (英語) の Tim Wong 氏は、次のように述べています。「フィーチャーフラグは、デプロイとリリースを切り離すためのものです。従来の企業では、この 2つはひとまとめになっており、デプロイが完了した時点でコードがリリースされます。フィーチャーフラグを使用すると、コードをデプロイするタイミングと、コードをリリースするタイミングをそれぞれ決定できます。実世界でテストしてから、ユーザーに提供する準備ができているかどうかを判断できるのです。これは非常に大きな違いです」

フィーチャーフラグのさらなる効果として、インシデント中に防御策として使用できることが挙げられます。たとえば、本番環境で問題が発生した場合、開発者はその原因となっている機能を無効にするだけで、修正が完了するまで問題のあるコードを効果的に隔離できます。通常、これは変更全体をロールバックするよりも迅速なため、全体的な安定性が向上します。

障害からの効率的なリカバリー

一連のテストが適切であっても、本番環境へのバグの流入は避けられません。バグが発生したときには、迅速で信頼性の高い回復プロセスを実施することが重要です。

Stripe が配信しているオンライン専門誌『Increment』が Amazon、Facebook、Google をはじめとする大手企業のインシデント対応について実施した調査の結果 (英語) から、ベストプラクティスは各社とも一致していることが明らかになっています。待機中のエンジニアへの通知、問題を一貫した方法でトリアージするための手順書など、大半の組織は本番環境が停止した場合の対応プロセスを明確に定義していました。メインラインブランチの正常性を維持するためには、こうした対応を規定して実践することが不可欠です。

本レポートに関連して注目したいのは、大手企業は解決の前に問題を軽減しているという点です。これを実践する場合は通常、根本原因を修正するのではなく、変更をロールバックします。エラーのデバッグには時間がかかり、そのままでは失敗状態が長引くのに対して、変更をロールバックすればバグの影響をすぐに抑制できます。

障害の代償に対する共通認識

バグは単なるコードベースのエラーではなく、皆様とユーザーの双方に多大な影響をもたらします。そうした影響を理解していれば、作業に優先順位を付け、技術的な問題を解決するモチベーションを高めることができます。

Code For America の 1部門である GetCalFresh では、エンジニアが毎日 1時間の時間を割いて、利用者がフードスタンプ (低所得者向けの食料費支援制度) の申請書を記入する状況をモニタリングしています。Code For America でかつてインフラストラクチャ担当者を務めていた John O'Duinn 氏は、この意図について次のように述べています。「政府の事務手続きは 5時に締め切られるため、あえてこの時刻にモニタリングすることにしました。この時間は作業に集中し、だれもが『人々に食料を届けたい』と考えています。この 1時間はプレッシャーとの戦いです」

この時間内に不具合が発生すると、エンジニアはそれを目の当たりにすることになります。抽象的な「エラー」を具体的に理解できるようになり、修正しようという意欲が高まります。エンジニアは自分の仕事の成果を確認できるだけでなく、問題が発生すれば人々の生活に影響することも実感できます。ただし、多少のプレッシャーは問題ありませんが、ミスを恐れる消極的な文化に発展してしまわないように注意が必要です。

潜在的な競合に関するコミュニケーション

障害を防ぐ最も確実な方法はテストですが、それ以外のところで見落とされがちなのが、コードの潜在的な競合に関する認識とコミュニケーションの欠如です。技術スタックの各パーツ (バックエンド、フロントエンド、運用など) に応じてチームを構成するやり方は理にかなっているように思えますが、そこには欠点があります。エンジニアをチーム分けすると、別

のチームの作業が見えづらくなり、自分の書いたコードが他の部分にどう影響するかを把握しにくくなります。

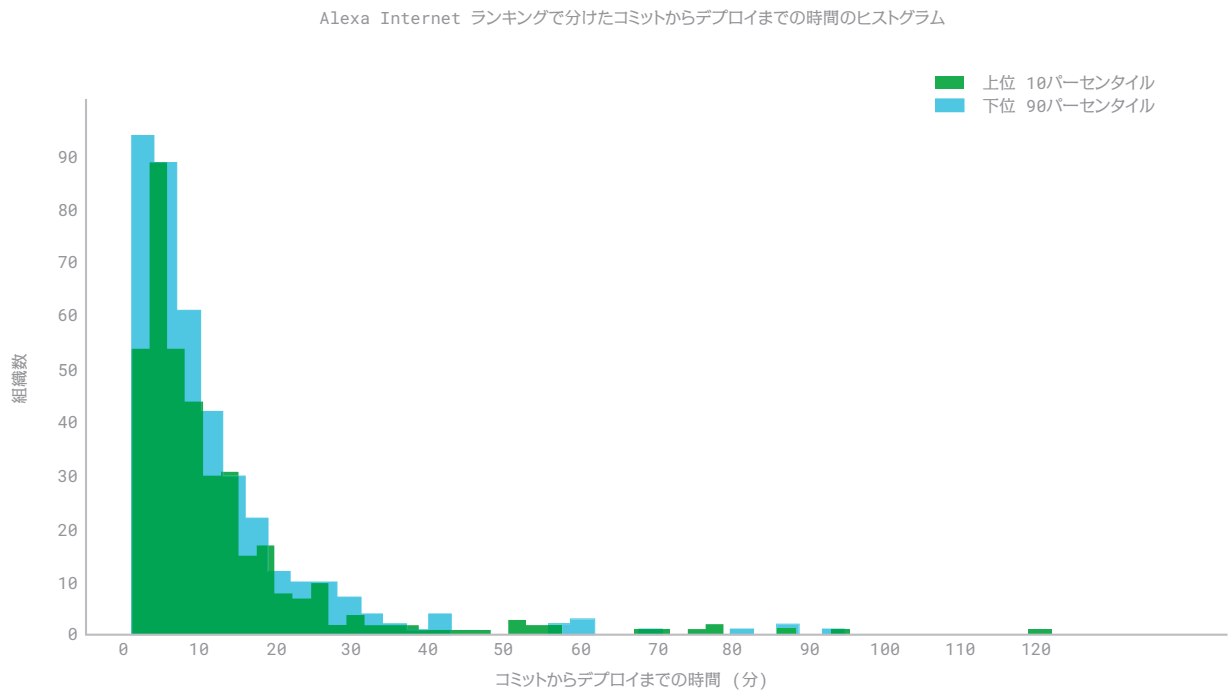
Code.org では、3 〜 7名のエンジニアから成る機能横断的なグループ「カバル (Cabal)」を編成しています。エンジニアリング責任者を務める Jeremy Stone 氏は、次のように述べています。「Code.org をミッション別に分割し、類似のプロジェクトに取り組んでいるエンジニアを 1つのグループにまとめることにしました。少人数のため、エンジニアの集中力に支障をきたすことなく、チームで毎日ミーティングを行えます」

各カバルはデイリースタンドアップを実施して、計画外の作業を共有し、潜在的な競合について確認しています。ただし、このようなデイリースタンドアップは、ゴルディロックス効果 (ちょうど良い程度を選択する心理現象) が働いてこそ有効です。参加するエンジニアの数は、少なすぎても多すぎてもいけません。生産的なミーティングに十分で、かつ他のエンジニアの状況を把握できる程度の人数にします。このようなコミュニケーションは、作業の重複や競合を防止するうえで重要です。作業の重複や競合は、メインラインブランチが失敗状態になる可能性を高める要因となるからです。

デプロイ時間

デプロイ時間は概ねコントロールされており、80.2 % の組織が 15分以内にデプロイを行っています。特にデプロイ時間の短い組織 (95パーセンタイル) では 2.7分、中央値は 7.6分です。それ以降はロングテールの分布が続き、下位 5パーセンタイルの組織のデプロイ時間は 30分です。

Alexa Internet ランキングの上位 10パーセンタイルの組織のうち 80 % が 17分以内にデプロイし、上位 5パーセンタイルのデプロイ時間は 2.6分です。これらの組織の中央値は 7.9分で、下位 5パーセンタイルのデプロイ時間は 36.1分でした。



このようにデプロイ時間が短い結果となったのは、これらのデータが CircleCI を利用している組織から取得したことが理由の 1つだと考えられます。もちろん贔屓目はありますが、それでも開発者が専用の CI/CD プラットフォームを使用することで、デプロイが完了するまでの時間を短縮できていることを嬉しく思います。

手動での QA からの解放

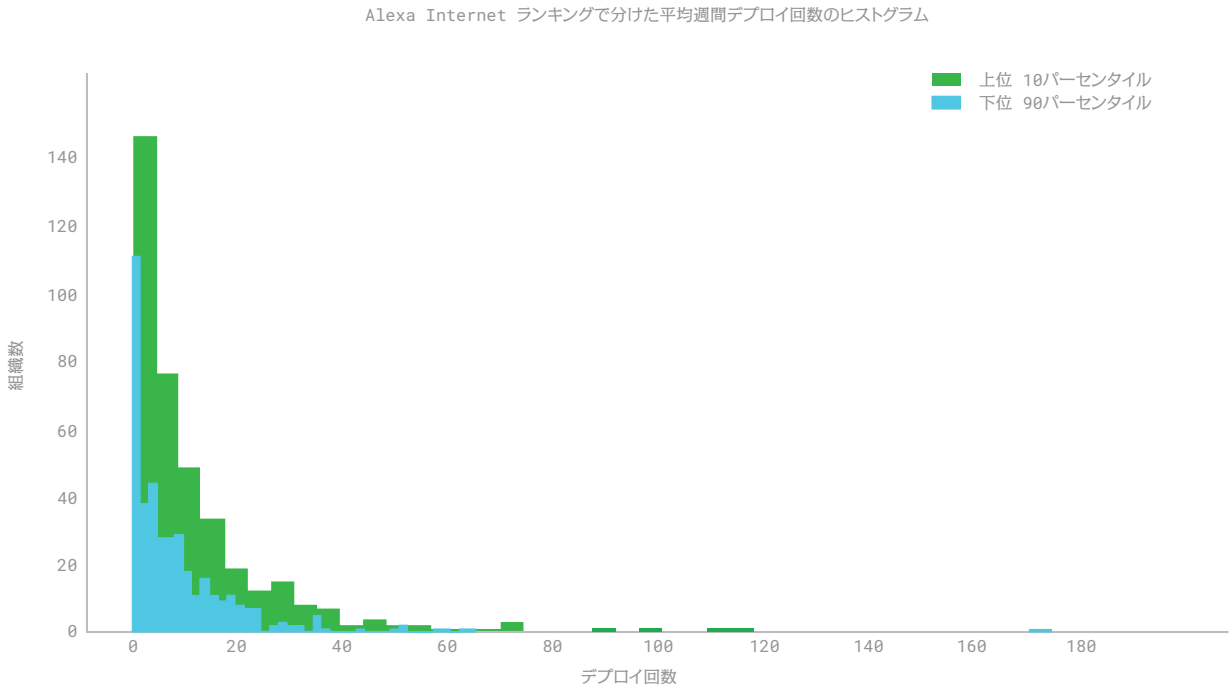
堅牢なテストを実施することの副次的効果として、手動での QA に費やす時間を削減できます。ただしそれには、組織がプロセスを十分に最適化して、バグをゼロに近づけること、障害から効率的に回復すること、継続的にモニタリングを続けることが必須条件となります。

これらの条件を満たしていれば、少なくとも安全性が確保される場合に、顧客を QA チームの一員として扱うことができます。ただし、これは QA が重要ではないという意味ではなく、コストのかかる手動での作業を可能な限り自動化することに明確なメリットがあるということです。

デプロイ頻度

全組織の 75 % が、最も活発なプロジェクトにおいて週に 12回以下の頻度でデプロイを行っていました。上位の組織 (95パーセンタイル) では、メインラインブランチを週に 32回デプロイしています。これは中央値の 5倍以上、下位 5パーセンタイルの約 24倍です。

Alexa Internet ランキングの上位 10パーセンタイルの組織については、右裾の伸びが長くてばらつきも大きく、95パーセンタイルに位置する組織では週に 42回デプロイを行っています。これは下位 5パーセンタイルの 40倍ですが、中央値 (週 8回) の 5倍というのは変わりません。ランキング上位組織のうち 75% が週に 15回以下の頻度でデプロイしており、ここにも差が表れています。



これら 3つの指標を見ると、Alexa Internet ランキングの上位の組織は特に変更が多いことがわかります。デプロイ頻度については、Alexa Internet ランキング上位の組織はそれ以外の組織よりもやや右側に分布しており、上位 10 パーセンタイルの組織がコードのプッシュ回数でも上位を占める傾向にあります。この結果はスピードの重要性を反映しています。あらゆるカテゴリで最も優れたパフォーマンスを上げるためには、デプロイのスピードも最大化する必要があるのです。

ベストプラクティス

簡潔かつ短期間のプルリクエスト

デプロイ頻度は複合的な指標であり、メインラインブランチの安定性とコミットからデプロイまでの時間 (CDT) の両方に依存します。よく知られている効果的な改善策は、プルリクエスト (PR) のサイズと期間を抑えることです。

Code.org の Jeremy Stone 氏は言います。「私たちの組織には、ブランチで長期間にわたって作業しないという文化が根付いており、現に約 80 % のブランチが 24時間以内にマージされています」

Code.org の主任ソフトウェアエンジニアの Brad Buchanan 氏がさらに詳しく付け加えます。「プルリクエストの中には 1、2行のコードという簡潔なものもあり、大部分は 200行以内に収まります」

このようなことは不可能に思われるかもしれませんが、この 2人のコメントからは、作業を小規模かつリリース可能な単位に分割するという Code.org の取り組みがうかがえます。こうした取り組みによって、Code.org はインクリメンタル開発の文化を築き上げ、より短期間のうちにコードレビューを完了してユーザーに届けられるようになりました。

フィーチャーフラグ

プルリクエストのサイズを抑えてデプロイ頻度を向上させるもう 1つの方法は、フィーチャーフラグの使用です。LaunchDarkly (英語) の Tim Wong 氏は、次のように述べています。「プルリクエストのサイズが大きくなる原因は依存関係です。依存関係を個別にリリースすることで、プルリクエストのコード量を削減できます」

作業を消化しやすいサイズに分割することは、継続的ソフトウェア開発の中核を成す考えです。フィーチャーフラグを使用すると、開発者は作業が完成した時点で順次リリースすることができます。開発者は作業を完了することで達成感が得られ、ユーザーは製品が変わらず機能することに満足します。

指標の相互関係

3つの指標を個別に紹介してきましたが、どれも独立した要素ではなく、どれか 1つを最適化すれば他にも影響します。たとえば、失敗状態の割合が 5 % 未満の組織では、デプロイ頻度の中央値は週に 5.3回、CDT の中央値は 6.7分になります。一方、それ以外の組織のデプロイ頻度の中央値は週に 8.7回、CDT の中央値は 11.7分です。

適切なテストを行わずに短期間で変更を繰り返すと、安定性が低下し、デプロイ頻度が高くなります。そのようにスピードのみを重視すると、ミスの修正に費やす時間が増える可能性があります。そう推測すると、CDT が長い組織ほどデプロイ回数が多い理由の説明がつきます。つまり、そのような組織は計画に沿っているわけではなく、必要に迫られてデプロイしていると考えられます。

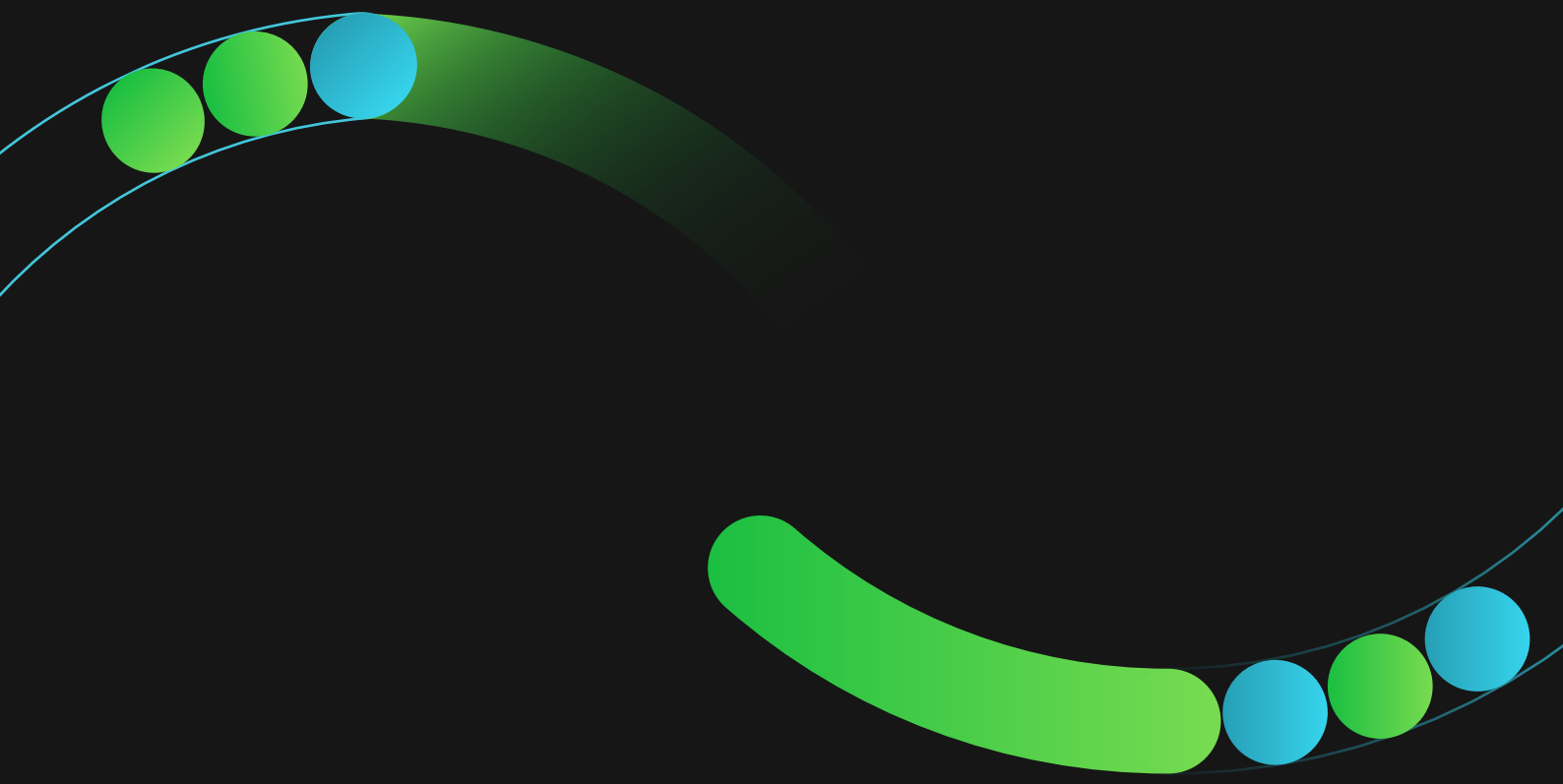
デプロイ頻度のみを優先させている組織では、粗悪なコードなどの技術的負債が増えるため、不安定な状態になる可能性が高くなります。また、バグの多いコードをリリースすれば、

問題を修正するためにデプロイ回数も増えます。このことを踏まえると、デプロイ頻度は組織のスピードを測る確実な指標としては扱えません。

ビルド時間が 12分以内の組織では、デプロイ頻度の中央値は週に 5.3回、不安定性の中央値は 0.2 % です。ビルド時間が 12分を超える組織では、デプロイ頻度の中央値は週に 8.3回、不安定性の中央値は 2.7 % です。この結果は、先ほどの推測を裏付けています。**お客様のニーズにすばやく対応しようと無理をすると、失敗状態にある時間やプロジェクトのビルド時間が増加する可能性があります。**

そのため、最適なバランスを見つけることが重要です。

デモグラフィック

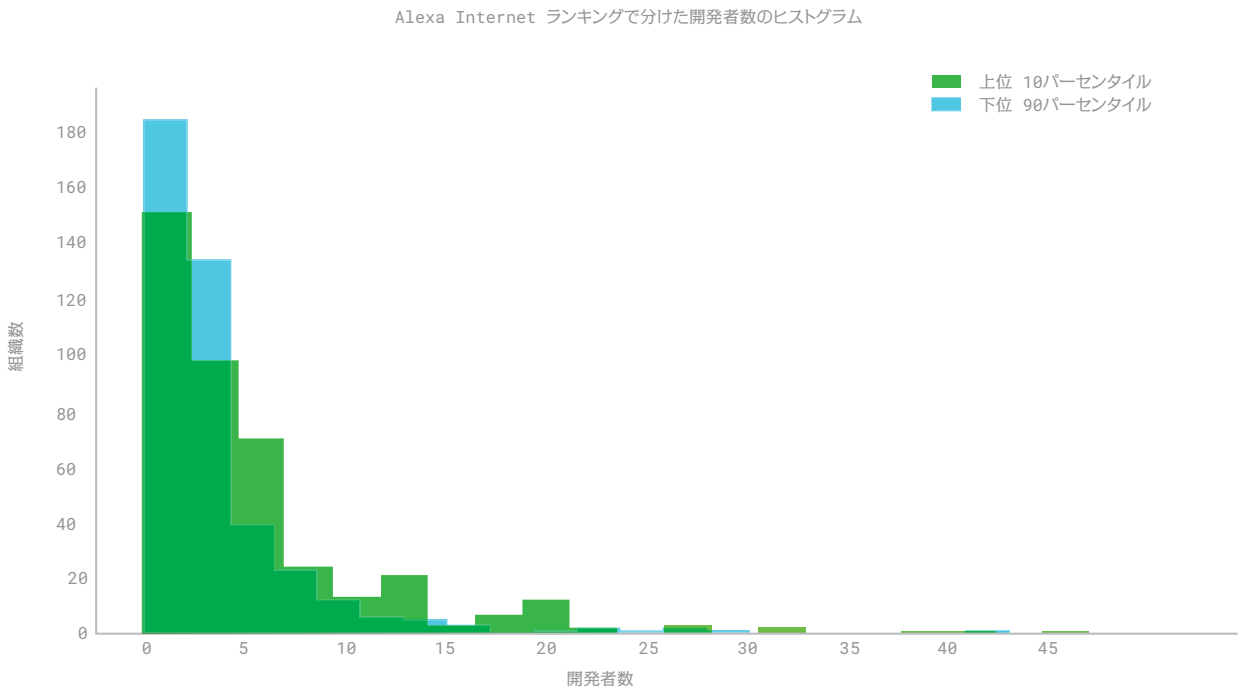


3つの主要な指標に加えて、地理的分布、CircleCI ビルドを実行するユーザー数、プログラミング言語など、複数のデモグラフィック変数についても調査しました。

地理的分布

サンプリングした組織のうち 79 % では、すべての開発者が同一の国で働いています。こうした単一国籍の開発チームの 56 % は、日本、英国、米国を拠点としています。Alexa Internet ランキングの上位 20パーセンタイルの組織については、74.2 % が単一の国を拠点としています。

この結果からわかるのは、成長のために CI を活用するうえでチームの地理的分布は関係ないということです。単一の国を拠点とするチームと、しっかり構成されている多国籍チームの間に大きな差はありません。メンバーが同じ部屋で働いていても、タイムゾーンを超えて散らばっていても、チームの成功にほとんど影響しないと自信を持つべきです。



ビルドを実行するユーザー数

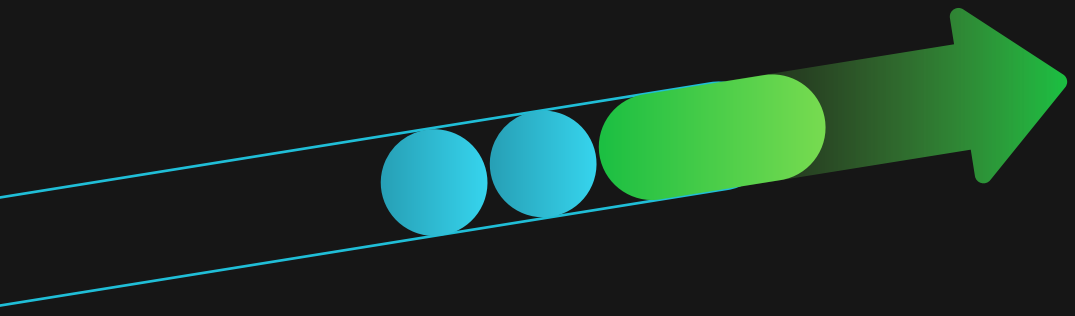
サンプリングした組織の 95.4 % では、週に平均 10人以下の開発者がビルドを実行しています。Alexa Internet ランキングの上位 10パーセンタイルの組織のうちの 85.3 % でも、これと同じ結果が得られました。

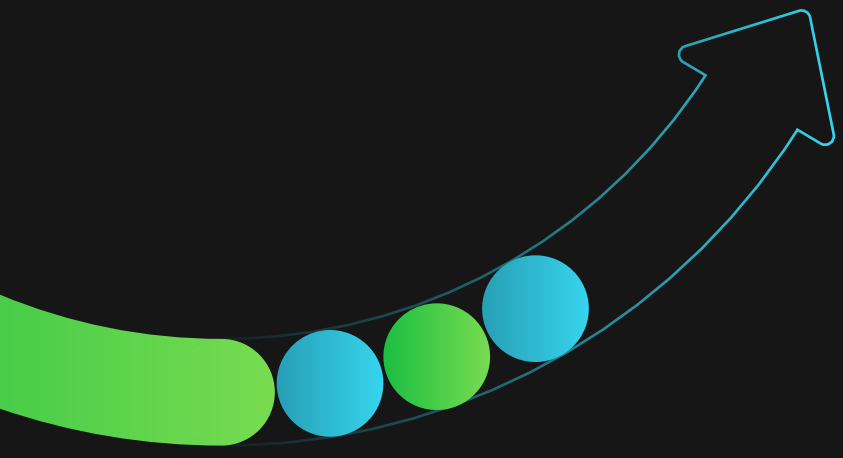
特に優れたパフォーマンスを上げている組織では、プロジェクトを小規模なチームに分割している可能性が高いと考えられます。前述の Code.org の例のように、小規模なチームの方が効果的に作業を分割し、緊密なコミュニケーションを維持できる傾向にあります。

主要なプログラミング言語

主要な言語が指定されているプロジェクトのうち、48.2 % は主に JavaScript または Ruby を使用しています。ただし、これら 2つの言語は、Alexa Internet ランキングの上位 10パーセンタイルの組織のプロジェクトの 49.3 % にとどまっています。この結果から、エンジニアの選択する言語が組織の成功に与える影響はごくわずかだと言えます。

まとめ





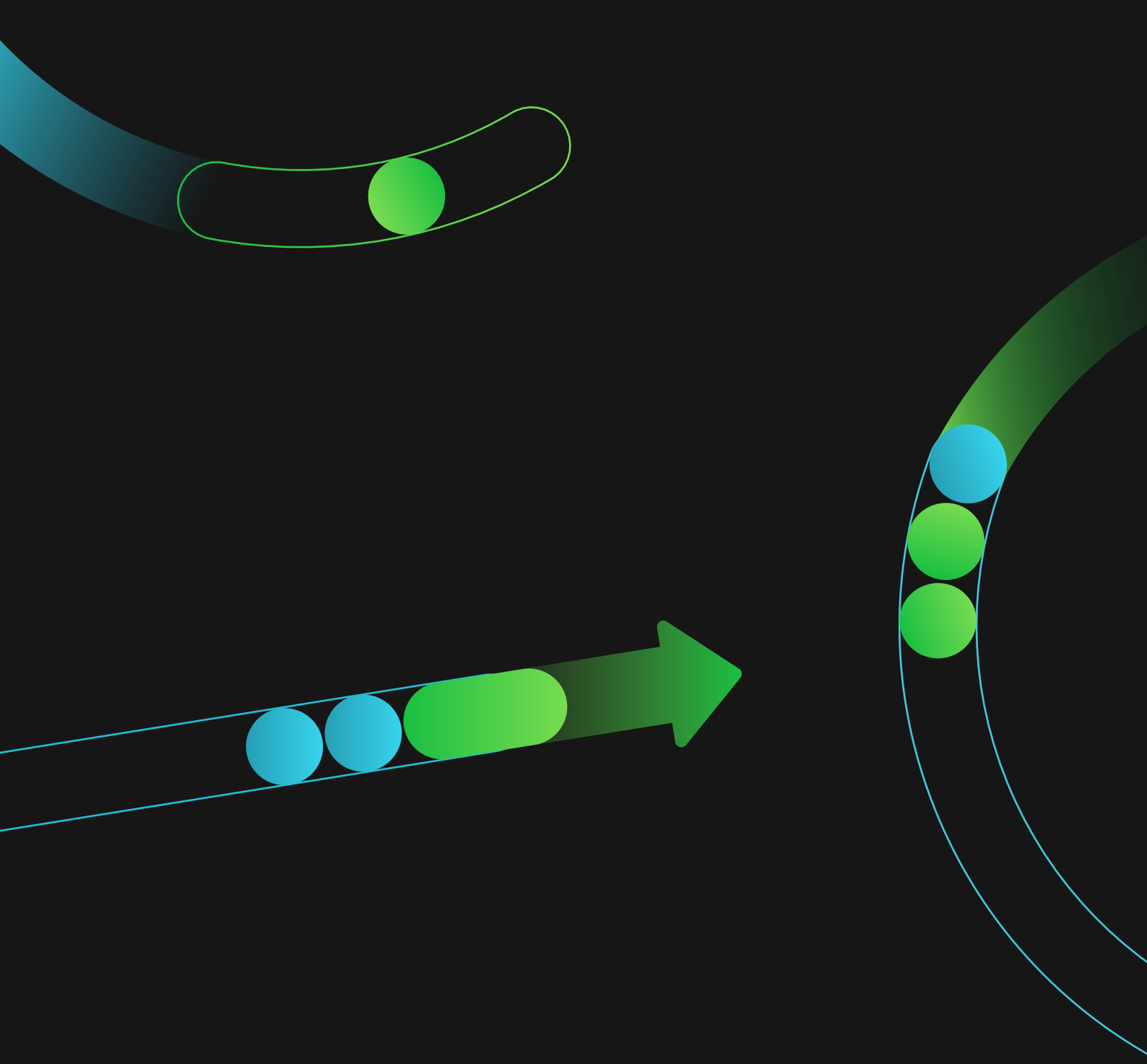
の中の変化のスピードは加速し続けています。その中で生き残るためには、組織は常にそうした変化に適応しなくてはなりません。DevOps の文化を取り入れるうえでは、組織を変革し、変化を実現することが求められます。絶え間なく変化する世界に合わせて自社を変えていく能力が、企業の成功を左右するのです。

しかし、「デジタルトランスフォーメーション」は一夜にして完成するものではありません。特に優れたパフォーマンスを上げている企業は、自社のソフトウェア開発パイプラインを戦略的投資、さらには企業価値と捉えており、継続的インテグレーションのベストプラクティスを絶えず組み込んでいます。

このような企業には、スループットの向上、エンジニアの満足度向上、顧客ニーズの的確な把握など、さまざまな成果が現れます。また他にも、組織が新しいアジャイルな開発哲学を採用するメリットは多数あります。今回の調査で明らかになった 3つの指標を最適化している組織は、成功に向けた準備が整っていると言えます。こうした先人のベストプラクティスに従うことで、皆様も DevOps の概念を理論から実践に移すことができます。

CircleCI がこれほどの規模で調査を実施したのは今回が初めてであり、興味深い結果が得られたと考えていますが、正確に理解できたかといえばそこまでの確証は持てません。とは言え、これまで未知だった領域に踏み込む最初の一歩となりました。今後も繰り返し調査を実施してまいります。当初の見解を伝えるところから始め、ユーザーの皆様や仲間との意見交換を続けていく予定です。今回得られた結果は今後の調査に組み込み、ソフトウェア開発の世界をさらに正確に把握できるよう前進していきたいと考えています。

皆様にもご協力いただければ幸いです。



circleci

今すぐビルドを始めましょう

www.circleci.jp