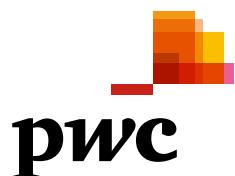


トラストとともに駆ける

—DevOpsにおけるコンプライアンス対応の要所





目次

はじめに	3
DevOpsとは	4
コンプライアンス対応とは	6
ケース設定	9
要件#1 職務の分離への対応	12
要件#2 テストと承認への対応	19
要件#3 アクセス権を必要最小限に保つ対応	22
推奨事項 可監査性の確保への対応	26
解釈を再铸造する	28
おわりに	29



はじめに

顧客向けか社内向けかにかかわらず、今やデジタルはビジネス活動の主幹です。そして、デジタル化したビジネスで成功している組織のほぼ全てにおいて、パフォーマンスを最大化するための優れた仕組みを見て取ることができます。一方で、ビジネスが拡大すると同時に、その利害関係者との関係は質・量ともに変化し、そこには必ず、規制当局やそれに類する関連団体によるコンプライアンス要件が存在します。

ビジネスパフォーマンスとコンプライアンス対応の両立は、今まで多くの組織における課題でした。要件にうまく対応できた組織は、利害関係者への説明責任を果たしながらも、パフォーマンスをより一層高め、競合他社に対するリードを確保できています。デジタルを取り入れたビジネスにおいては、特にIT関連のコンプライアンス要件に注意深く対応する必要があります。なぜなら、その対象であるプログラム開発やシステム運用などの手法が、ビジネスパフォーマンスの源でありながら、昨今の技術進展に伴い急激に進化しているからです。

「DevOps」と呼称される概念の根幹には、数々の先進的な手法が束ねられています。そして、その先進性ゆえに、コンプライアンス対応上の厄介な論点が存在します。

- ・ビジネスを拡大するのであれば、DevOpsを諦めなければならないのでしょうか？
- ・もし諦めないとしたら、コンプライアンス要件にどのように向き合うべきでしょうか？

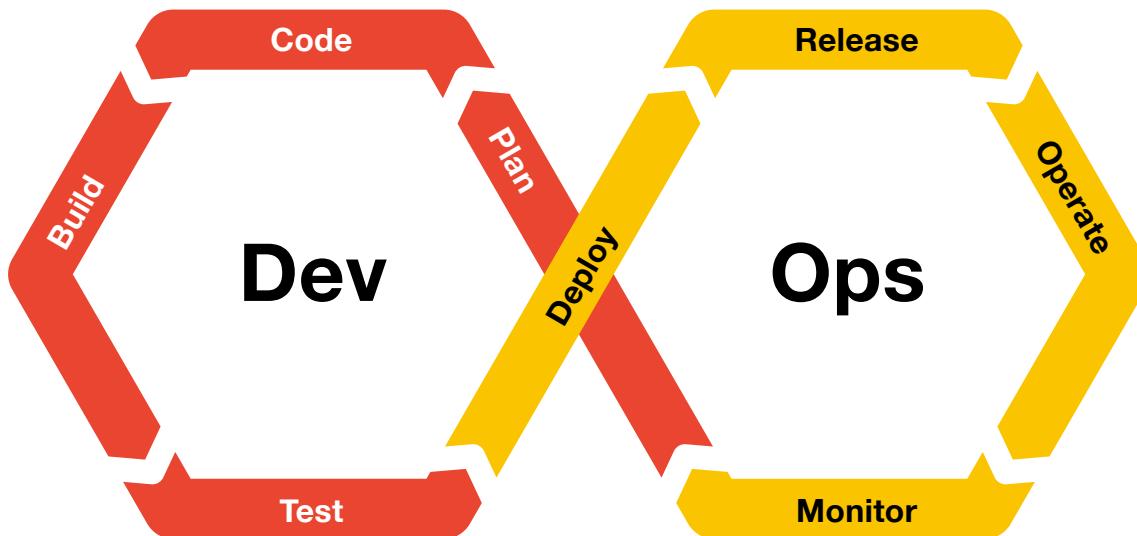
本レポートでは、DevOpsを採用する架空企業の具体的なケースを用いて、ビジネスパフォーマンスを最大化・最適化し続けながら、コンプライアンス要件とどのように向き合うべきかについての参考となる情報を提供します。

なお本レポートは、PwCあらた有限責任監査法人と株式会社永和システムマネジメントが共同で作成しました。

DevOpsとは

DevOpsとは、一般的に「ビジネスの価値をより確実かつ迅速にエンドユーザーへと届け続けるために、開発チームと運用チームが協調してソフトウェアサービス／システムを開発する環境やプロセス、組織文化」と理解されています¹。米国政府のDigital.gov²にもDevOpsコミュニティが組成されるなど、近年、大規模な組織にもDevOpsが浸透しています。

図表1：DevOpsの概念



1 : DevOpsの原点とされる「Velocity 2009」というイベントにおいて、John Allspaw氏とPaul Hammond氏が発表した「10+ Deploys Per Day: Dev and Ops Cooperation at Flickr」を参考にした。このプレゼンテーションの内容は資料が公開され、DevOpsエンジニアに広く認知されている。

2 : GSA (U.S. General Services Administration : 米連邦政府一般調達局) のTTS (Technology Transformation Services : 技術革新サービス部門)

DevOpsを適用する組織は、従来と比較して、価値の提供・改善のスピードが飛躍的に向上し、柔軟性も実現されています。

また、「DevOps」をキーワードに、さまざまな原則や優れたプラクティスに関する情報が発信されており、例えば『The DevOps HANDBOOK³』では「3つの道」が示されています（図表2）。

図表2：DevOps「3つの道」

第1の道： フローの原則	第2の道： フィードバックの原則	第3の道： 継続的な学習と実験の原則
開発から運用、そして顧客に仕事の成果を送り届けるまでの流れを加速する	高品質・高信頼性・安全性を実現するために、開発・運用の各ステージや顧客からの全ステージで、素早くコンスタントにフィードバックを得る	日常的な業務改善のための学習・実験の制度や組織文化を作る

本レポートでは、第1の道「フローの原則」に含まれる以下の事項に注目し、これを実践する組織が特定のコンプライアンス要件を達成するための考慮事項を、具体的なケースを挙げて記述しています。

- ・デプロイパイプラインの基礎を作る
- ・高速で信頼性の高い自動テストを実現する
- ・継続的な統合とテストを実現する
- ・ローリスクリリースを実現する自動化、手法、アーキテクチャを実装する

図表3：DevOpsで実現された世界

企業	DevOpsによるパフォーマンスの実現
消費財	同社では、販売サイトの頻繁な更新が求められる一方で、人気の新作商品を発売するたびにサイトはクラッシュしていた。DevOpsを採用し大規模な変革を行ったところ、本番環境へのデプロイをする際に6週間を要していたものが、1日に3回のデプロイが行われるようになり、サイトのクラッシュも発生しなくなった。
エンタメ	同社では10年近くにわたって積極的にDevOpsを追求しており、その成果の1つがビデオストリーミングサービスである。ユーザーのニーズを満たすソフトウェア開発を高速かつ継続的に実現し続けてきた結果、ブランド力も相まって、すでに確立されていた競合サービスにとっての本格的な競争相手となり、サービス開始後わずか1年弱で1億人の加入者を達成した。
EC	当該ECサイトおよび関連のサービスでは、デプロイの仕組みを劇的に改善した。年間50万回、毎秒1回のデプロイを可能にして、ビジネスニーズを満たす高速かつ信頼性のあるソフトウェア開発を実現した。
航空機	数百億米ドルの売上高を持つ同社では、信頼性は最優先である。推定数千万行のコードを相手にDevOpsを導入することは非常に困難であった。しかし、「CI（継続的な統合）」の採用は画期的な成果を達成し、現在、毎日1万5,000回の自動テストプロセスを実行している。

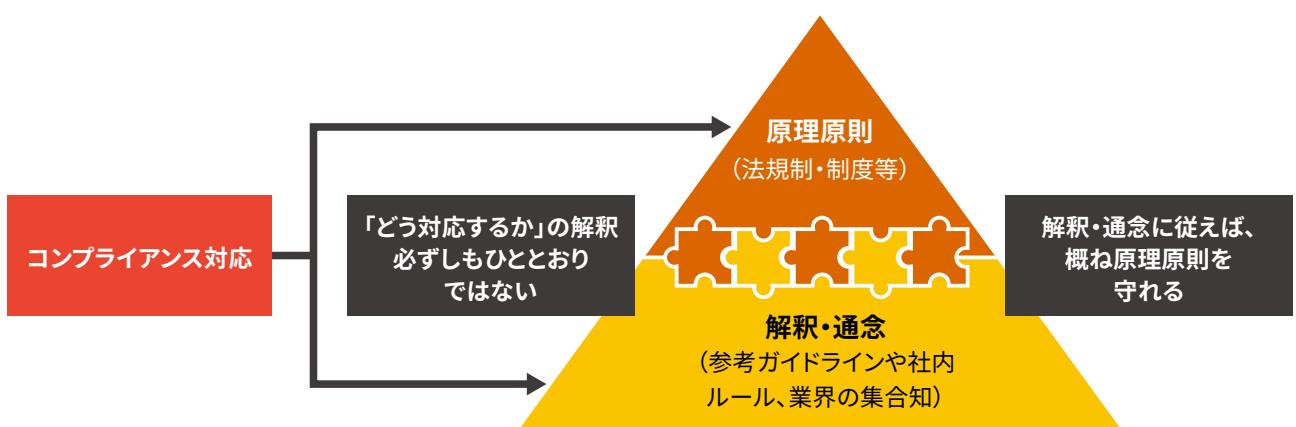
3 : Gene Kim, Jez Humble, Patrick Debois, John Willis, 2016. The DevOps HANDBOOK: It Revolution Pr



コンプライアンス対応とは

DevOpsにおけるコンプライアンス対応の要所を検討するにあたり、前提となる「コンプライアンス対応」について言及します。「コンプライアンス対応」の理念型は、その根本となる原理原則と、実践する上での解釈・通念に識別することができます。

図表4：原理原則と解釈・通念



原理原則

特定の領域で活動する以上遵守すべき社会のルールであり、ハードなものでは法規制、ソフトなものであれば認証制度等がこれに相当します。原理原則は遵守が必要なルールであり、これに反すると、罰則や市場からの退場に発展します。

例として、日本の上場会社の内部統制報告制度を取り上げます。

解釈・通念

原理原則を達成する方法についての一般的・伝統的な具体的理解です。各種業界の（必達ルールではなく参考の位置づけである）ガイドライン、社内ルール、業界人の集合知における特定領域のチェックリスト等がこれに該当します。原理原則を解釈した結果形成されたものですので、解釈・通念に反しなければ、原理原則を概ね守れます。原理原則を達成する方法が必ずしも伝統的解釈だけであるとは限りません。

内部統制報告制度におけるITの統制の概要⁴

- ・財務報告の信頼性を確保するためのITの統制は、会計上の取引記録の正当性、完全性及び正確性を確保するために実施される。正当性とは、取引が組織の意思・意図にそって承認され、行われることをいい、完全性とは、記録した取引に漏れ、重複がないことをいい、正確性とは、発生した取引が財務や科目分類などの主要なデータ項目に正しく記録されることをいう。
- ・金融商品取引法による内部統制報告制度においては、ITの統制についても、財務報告の信頼性を確保するためには、ITの整備するものであり、財務報告の信頼性以外の他の目的を達成するためのITの統制の整備及び運用を直接的に求めるものではない。
- ・ITを利用した情報システムにおいては、一旦適切な内部統制（業務処理統制）を組み込めば、意図的に手を加えない限り継続して機能する性質を有しているが、例えば、その後のシステムの変更の段階で必要な内部統制が組み込まれなかつたり、プログラムに不正な改ざんや不正なアクセスが行われるなど、全般統制⁵が有効に機能しない場合には、適切な内部統制（業務処理統制）を組み込んだとしても、その有効性が保証されなくなる可能性がある。こうした問題に対応していくためには、例えば、
 1. システムの開発又は変更に際して、当該システムの開発又は変更が既存のシステムと整合性を保っていることを十分に検討するとともに、開発・変更の過程等の記録を適切に保存する
 2. プログラムの不正な使用、改ざん等を防止するために、システムへのアクセス管理に関する適切な対策を講じるなど、全般的な統制活動を適切に整備することが重要となる。

衝突しがちなコンプライアンス対応

解釈・通念にはDevOpsと衝突する内容が多く存在します。例えば、

- ・開発担当者と運用担当者を職務分離すること：DevOpsでは開発と運用の壁を取り払うことで、価値提供スピードを高めます。一方で、開発担当者と運用担当者の分離は最も基礎的なコントロールであり、コンプライアンス要件として求められます。
- ・設計書、ソースコード、テスト成果物など、工程ごとに上長承認を行うこと：DevOpsでは自動化されたパイプラインおよび自動テスト、ピアレビュー、手動テストの組み合わせによって価値提供スピードと品質が確保されます。一方で、コンプライアンス要件としては、定められた役職者によるテスト結果の承認行為が求められます。

特に1点目の「開発担当者と運用担当者の分離」は、「開発者と運用者の垣根を取り払う」DevOpsと、字義上は真っ向から対立します。

なぜ今、IT領域の解釈・通念がぶつかるのか

コンプライアンス対応における解釈・通念は、新たに登場あるいは改訂された原理原則の適用時に形成されます。IFRSの収益認識基準、各国のデータ保護規則、ESGにおけるEUタクソノミーなど、さまざまな領域で原理原則が適用され、遵守のための解釈がそのタイミングで発生します。

ITに観点を移すと、解釈を含む原理原則は1990年前後から複数登場しています。当時のIT実装目的は「大量の業務を正確かつ効率的に処理する」が主流であり、構成要素のIT製品・サービスや開発・運用のやり方もそれを支えていました。当然、コンプライアンス要件の伝統的な解釈・通念も、この実装⁶を考慮して形成されています。他方、近年登場したデジタルを主幹としたビジネスでは、新しいIT製品・サービスを駆使して「市場の変化を理解し、柔軟に適応する」実装が実現されています。この実装目的の違いは、開発・運用の実施方法の違いにも顕著に表れ、結果として1990年代～2000年代に形成された伝統的な解釈・通念と衝突しやすくなっています（図表5）。

4：金融庁 企業会計審議会「財務報告に係る内部統制の評価及び監査の基準並びに財務報告に係る内部統制の評価及び監査に関する実施基準の改訂について（意見書）」、2019

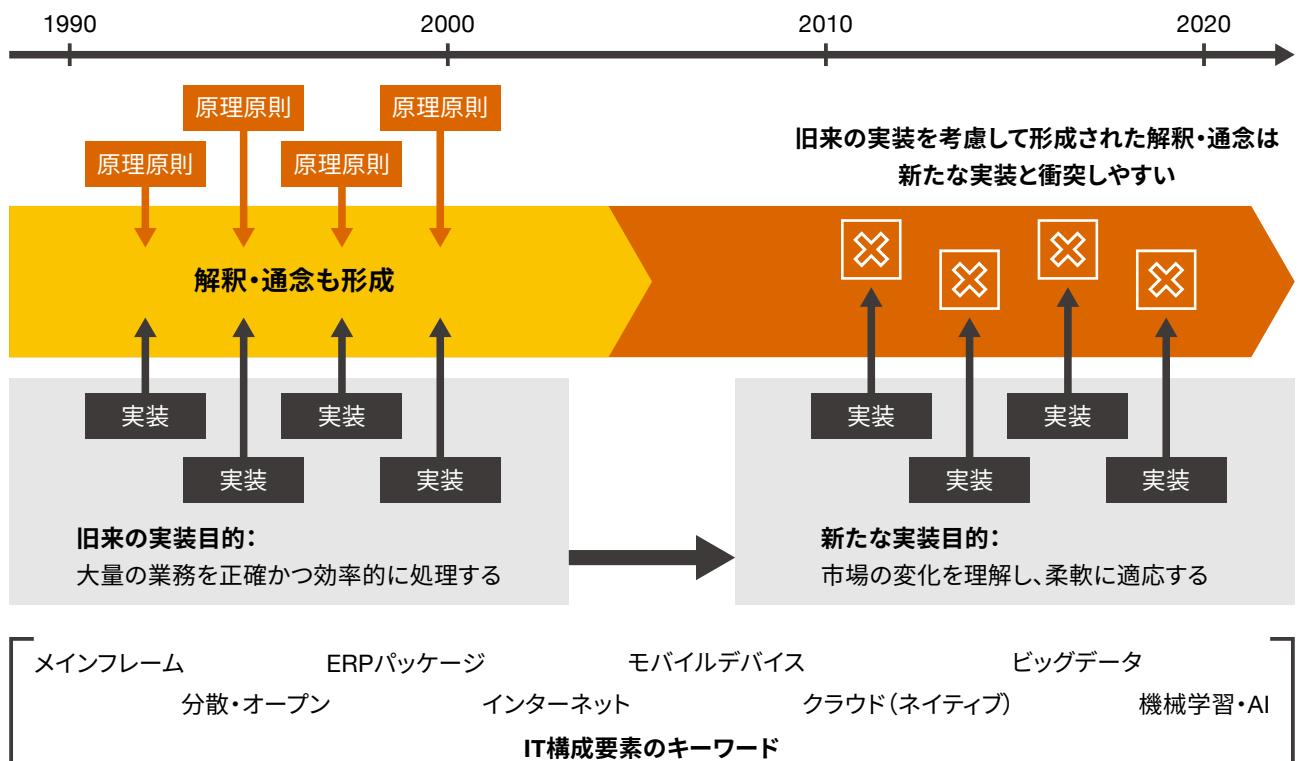
5：「ITの全般統制」を指しており、業務処理統制が有効に機能する環境を保証するための統制活動を意味し、通常、複数の業務処理統制に関する方針と手続を言う。具体例としては、システムの開発、保守に係る管理、システムの運用・管理、内外からのアクセス管理などシステムの安全性の確保、外部委託に関する契約の管理が挙げられる。

6：本文脈での「実装」は、ビジネス・業務全範囲の具体的なアーキテクチャ設計を指す。情報システムの構成要素（ハードウェアやソフトウェア、クラウドサービス等）やシステム開発・運用方法に加えて、業務オペレーションそのものも含まれる。

一方で、厳しいコンプライアンス要件に対応しながら、ビジネスパフォーマンスをさらに伸ばす組織は確かに存在しています。DevOpsでビジネスパフォーマンスを実現する組織においては、この衝突にどのように対応するのでしょうか。

以降では、拡大するプロダクトがステークホルダーからITコンプライアンス対応を求められるケースを例に、衝突が起こりやすい伝統的解釈とそれに対する具体的手段を検討します。

図表5：解釈・通念の形成と衝突





ケース設定

伝統的解釈との衝突と、その解消の方向性を検討してみましょう。ここでは、架空企業「AIファイナンスローン社」のケースを用います。

なお、本ケースで示される原理原則や実装は例示であり、個別具体的な組織がこれをそのまま採用することで、特定の原理原則に対応できるわけではありません。一般的に、その組織や内外環境のリスク要因が多いほど全体的なリスクは高くなる傾向にあり、原理原則に応じるための実装にも工夫や追加的措置が必要となります。以下に示された内容は、特定の原理原則に対応するための包括的なリストとしては使用できません。

ビジネスとサービス

AIファイナンスローン株式会社（以下A社）は、住宅・ファイナンスローンを取り扱う資金業者です。「プロダクトZ」と名づけられた、AI信用スコアエンジンを用いた自社開発のローン受付・審査・契約サービスによって、一般的なローン業者と比較してはるかに短時間で融資可能であることが強みです。

短時間融資を支えるシステム（申込受付Webサイトおよび審査契約バックシステム）の開発・保守と運用は、「市場の期待に圧倒的スピードで対応する」という思想のもと、DevOpsで実施されています。ローン商品開発からシステム開発・保守までを、プロダクトZ部門が一手に担っており、この価値提供スピードがA社のコアコンピタンスとなっています。

直面しているコンプライアンス要件

A社のビジネスが順調に拡大した結果、「大手金融グループによる友好的買収」が行われました。大手金融グループの一員となったA社は、その顧客基盤とブランド力を活かし、プロダクトZのシェアをさらに拡大しつつありました。

一方で、親会社の内部監査部門によるIT監査を受けたところ、IT統制について改善指摘が発生しました（図表6）。この改善指摘は、DevOpsの思想・実装と相反するものばかりのように見えます。

図表6：IT統制に対する改善指摘

#	原理原則	伝統的解釈を基にした実装の方向性
1 職務の分離	アプリケーションやインフラを不適切に変更できないように、職務を分離すること	<ul style="list-style-type: none"> ・アプリケーションの開発担当者と、本番環境のデータ修正やインフラ変更、リリース作業を行う運用担当者を分離する
2 テストと承認	データやアプリケーション、インフラへの変更が意図したとおり行われるよう、適切なテストと承認を行うこと	<ul style="list-style-type: none"> ・開発計画、本番作業計画について上長承認を行う ・設計書、ソースコード、テスト成果物など工程ごとに上長承認を行う ・変更内容のリリース前に上長承認を行う
3 アクセス権を必要最小限に保つ	データに対する直接的で不適切な変更ができないように、インフラへのアクセス権を必要最小限に保つこと	<ul style="list-style-type: none"> ・本番環境のインフラ（データベースやデータファイルを含む）へのアクセス権は本番作業直前に付与し、作業後すみやかに剥奪すること ・インフラへのアクセス権を設定できる特権IDの使用状況を監視すること ・インフラへのアクセス権が必要最小限であるかを定期的に確かめること

また、親会社内部監査部門からは以下のコメントも挙がっています。

IT統制に対する推奨事項

- ・IT統制を構成するルールや手順が一定程度存在するものの、そのとおりに統制が運用されているかを確認することが可能な証跡（ログ等）が記録・保全されていません。つまり、可監査性が不足しています。
- 特に、開発リーダーとプロダクトオーナーがリリース承認を判断する対象インクリメント⁷について、一連の承認されたPBIや作業チケットと確かに紐づいていることや、そのインクリメントが過不足なく確実にクラウド上の本番環境へリリースされていることを説明するためには、チケット承認履歴、構成管理ツールのコミット履歴、CI／CD⁸ツールのワークフロー実行履歴が整合していることが望されます。しかしながら、チケット承認やコミット履歴の記録（入力）漏れがあることから、チャットツール上の履歴で補足する必要がある状態です。
- ・ビジネスが拡大するにつれて、さまざまな監査／評価を受ける機会が増えると想定されますので、「ルールどおりに運用されている」ことを効率的に説明するために、関連するログの記録と保全をお勧めします。
- ・また、これらログ等を事業部門が自ら分析し、ルールから逸脱した運用を発見し、必要な処置を行うといった形で、発見的統制を強化することも可能です。

7：インクリメントの語義は「増分」であるが、アジャイル開発の文脈では「プロダクトゴールを達成するための具体的な更新内容」を意味する。

8：ソフトウェア開発において顧客価値の提供頻度を高めるために、継続的なインテグレーション（CI:Continuous Integration）と継続的なデリバリー（CD: Continuous Delivery）を実施する思想、作業習慣を指す。継続的インテグレーションでは、ビルドやテストを頻繁に繰り返すことで開発の効率化などを図る。継続的デリバリーでは、テストされたコード変更を自動で共有レポジトリにアップロードし実稼働環境にリリースできる状態とすることで、本番環境への反映の頻度を高める。

プロダクトZ部門

A社にとって、DevOpsは目的ではなく手段です。ビジネスパフォーマンスを最大化するため、プロダクトZ部門は「重要な仕事に集中し、価値提供スピードを最大化する」ことを表明しています。部門として何かを検討・判断する際にもたびたびこの表明の言葉が各部員から発言されます。

プロダクトZ部門は、今回のIT統制に対する改善指摘にも、この言葉に基づいて対応しようとしています。ビジネスを拡大するためにコンプライアンス要件への対応が必須と理解しているものの、その具体的な実装方法が価値提供スピードに直結する内容であることから、プロダクトZ部門全員が慎重になり、悩んでいます。

システムアーキテクチャ

プロダクトZはパブリッククラウド上に存在し、仮想サーバーやストレージ、各種マネージドサービスによって構築されています。監視やアクセス権管理にも、同クラウドのマネージドサービスを活用しています。

開発の流れ

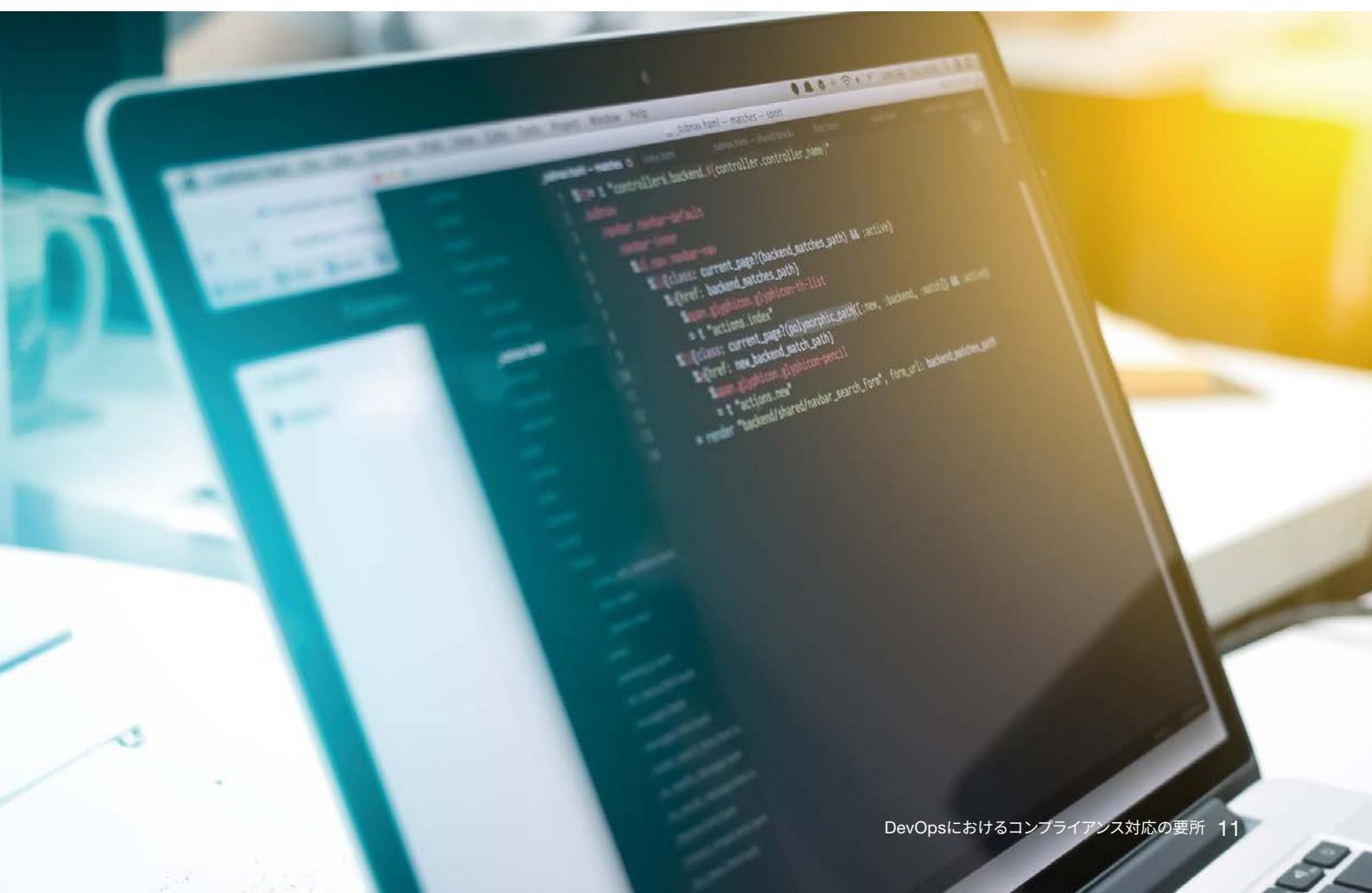
プロダクトZは2週間に1回のリリースが基本です。変更を伴う企画案は、管掌役員Aを含む部員全員によって検討され、「次回何を作るか」も、Aを含む部員全員で意見を出し合い決定します。

「何を作るか」を決定すると、管掌役員Aを除く開発メンバー、すなわち部員全員が作業を計画します。必要に応じて設計文書を作成・更新しますが、全てのリリース対象について作成するわけではなく、コーディングをすぐに行う場合もあります。

開発リーダーが「必要な作業が全て行われたこと」を、POが「リリースにあたって妥当なタイミングであること」を確認し、リリースされます。

開発環境、パイプライン

開発パイプラインは、構成管理ツールやCI/CDツールを中心としたツールチェーンによって構築されています。また、インフラ設定はコード化(IaC)されており、同パイプラインを使用して設定変更をリリースします。



要件#1 職務の分離への対応

アプリケーションやインフラを容易に変更できないように、職務を分離する

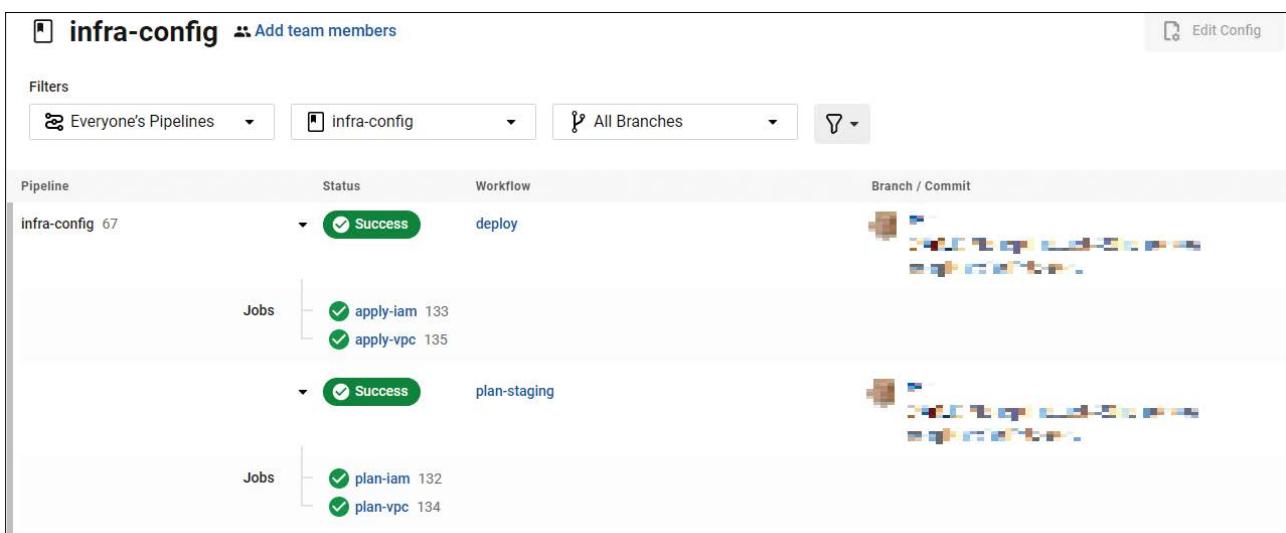
現在の状態

プロダクトZ部門には30名程度が在籍しており、多くの裁量が与えられています。具体的には、以下の作業を単独で実施できる権限を持っています。

- リポジトリに保存されたソースコードへのアクセス
- コーディング、各ブランチへのプルリクエスト、マージ
- Mainブランチから本番環境へのデプロイ／リリース
- 構成管理やCI／CDツールの設定変更

一方で、パブリッククラウド上の本番環境への直接アクセス権を常時保有している部員はいません。必要な際には、IaC上で権限設定に該当するソースコードを更新し、当該IaCをリリースすることで、権限を一時的に付与します。作業終了後、権限設定を元に戻す（剥奪する）リリースが行われます。

図表7：CI／CDワークフロー（インフラの更新）



The screenshot shows a CI/CD pipeline named 'infra-config' with 67 commits. The pipeline has a 'deploy' workflow. It consists of two stages: 'apply-iam' (133 runs, success) and 'apply-vpc' (135 runs, success). Below these is another stage 'plan-staging' with 'plan-iam' (132 runs, success) and 'plan-vpc' (134 runs, success). The interface includes filters for 'Everyone's Pipelines', 'infra-config', and 'All Branches', and a 'Edit Config' button.

※画面はイメージです。

図表8：権限設定のコード例

```
# group for developers
resource "aws_iam_group" "developers" {
  name = "sample_developers"
}
# group for operators
resource "aws_iam_group" "operators" {
  name = "sample_operators"
}
# group for auditors
resource "aws_iam_group" "auditors" {
  name = "sample_auditors"
}
# for developers on development environment
resource "aws_iam_group_policy_attachment" "developers_policy_PowerUserAccess" {
  group = aws_iam_group.developers.name
  policy_arn = "arn:aws:iam::aws:policy/PowerUserAccess"
}
resource "aws_iam_group_policy_attachment" "developers_ReadOnlyAccess" {
  group = aws_iam_group.developers.name
  policy_arn = "arn:aws:iam::aws:policy/ReadOnlyAccess"
}
resource "aws_iam_group_policy_attachment" "developers_IAMUserChangePassword" {
  group = aws_iam_group.developers.name
  policy_arn = "arn:aws:iam::aws:policy/IAMUserChangePassword"
}
# for operators
resource "aws_iam_group_policy_attachment" "operators_ReadOnlyAccess" {
  group = aws_iam_group.operators.name
  policy_arn = "arn:aws:iam::aws:policy/ReadOnlyAccess"
}
resource "aws_iam_group_policy_attachment" "operators_IAMUserChangePassword" {
  group = aws_iam_group.operators.name
  policy_arn = "arn:aws:iam::aws:policy/IAMUserChangePassword"
```

※インフラ環境構築のためのコード管理ツールの1つである「Terraform」を利用し、私たちで作成したサンプルコード。

原理・原則とのギャップ

このままでは、プロダクトZ部門の全員が「アプリケーションやインフラを容易に変更」可能であるため、何らかの対応が必要です。原理原則⁹に立ち返るために、極端なリスクシナリオを想定しました。

リスクシナリオ

プロダクトZ部門の賞与額はローン成約高と連動している。ある部員が賞与の増額を動機として、ローン成約高を操作することを企図した。この部員は日々の開発・運用作業から、ソースコードのどこをどのように変えれば、融資判断に用いる信用スコアが変化するかを熟知している。信用スコアエンジンとの内部インターフェースに不正な変更を加えることで、一定の条件下で信用スコアが高く表示・記録されるよう変更することに成功した。結果、高リスクの債務者に高い信用スコアが表示され、不適切な融資が実施された。

プロダクトZ部門は、DevOpsの基本思想を守りながら、このリスクシナリオを防止あるいは発見できる「アプリケーションやインフラを容易に変更できないように職務を分離する」統制を検討します。

両立させるための新たな統制

プロダクトZ部門は「要件#1」に対応するため、以下を実装します。

1. プランチ保護

構成管理ツールの設定として、各プランチへのマージ時には、プルリクエストを行った本人以外の部員による承認が必要となるよう設定する。

2. パイプライン保護

パイプラインを構成するCI/CDツール、構成管理ツールの設定の変更権限は、開発リーダーのみが持つこととする。

3. リリースを含む重要な行為の通知と確認

社内チャットツールに監視用チャンネルを作成し、Mainプランチへのマージ、本番環境へのデプロイパイプラインの動作、ツール設定の変更については、都度通知が挙がる設定とする。挙がった通知については、日次のスタンドアップミーティングで妥当であることを2名以上が確認し、確認し、コメントを残すこととする。申請漏れ等疑義がある場合には、チケットを起票し、開発リーダーの承認を得る。

4. 通知先である社内チャットツールの特権は別部門が保有

これにより、変更を行う部員とその内容を確認する部員の複数名が関与しない限り、データやアプリケーション、インフラを変更できない状態となりました。また、開発リーダー単独での実行が可能となるよう各ツールの設定変更を試みても、チャットツール上に変更の通知が挙がることで不審な行為を発見できます。さらには、この発見的統制¹⁰の有効性を確保するため、チャットツールのログ編集や保全設定の変更が可能な特権を、プロダクトZ部門外に移管しました。

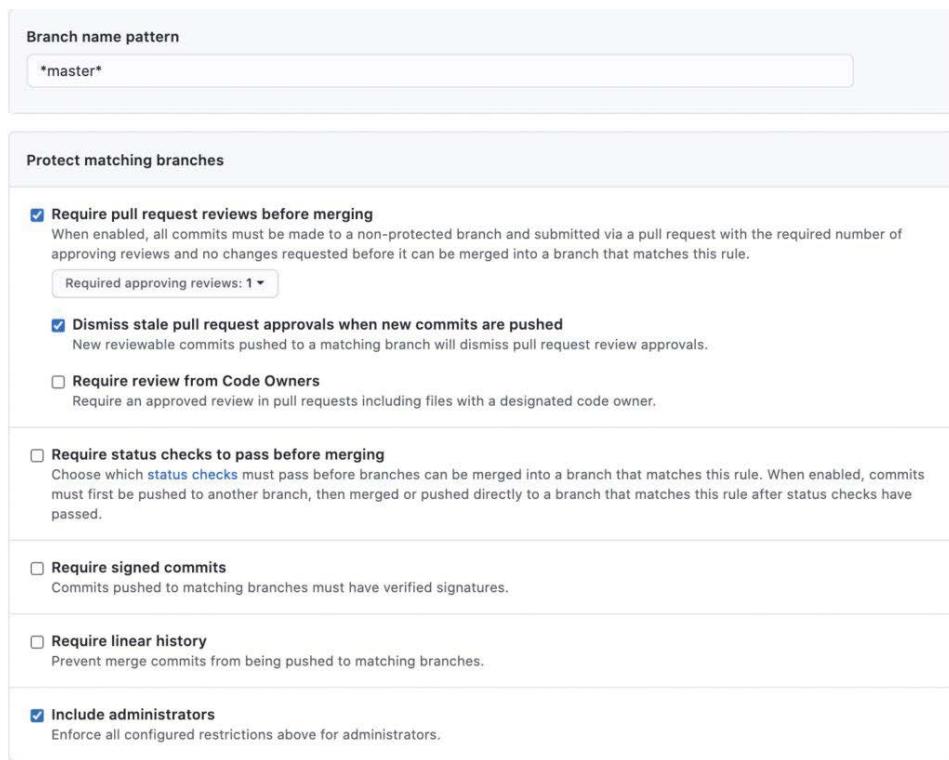
9:「財務報告の信頼性を確保するためのITの統制は、会計上の取引記録の正当性、完全性及び正確性を確保するために実施される」(P7「内部統制報告制度におけるITの統制の概要」参照)

10:発見的統制とは、財務報告の信頼性を損なう結果となり得る、すでに発生した誤謬や不正を発見することを目的とした統制を指す。

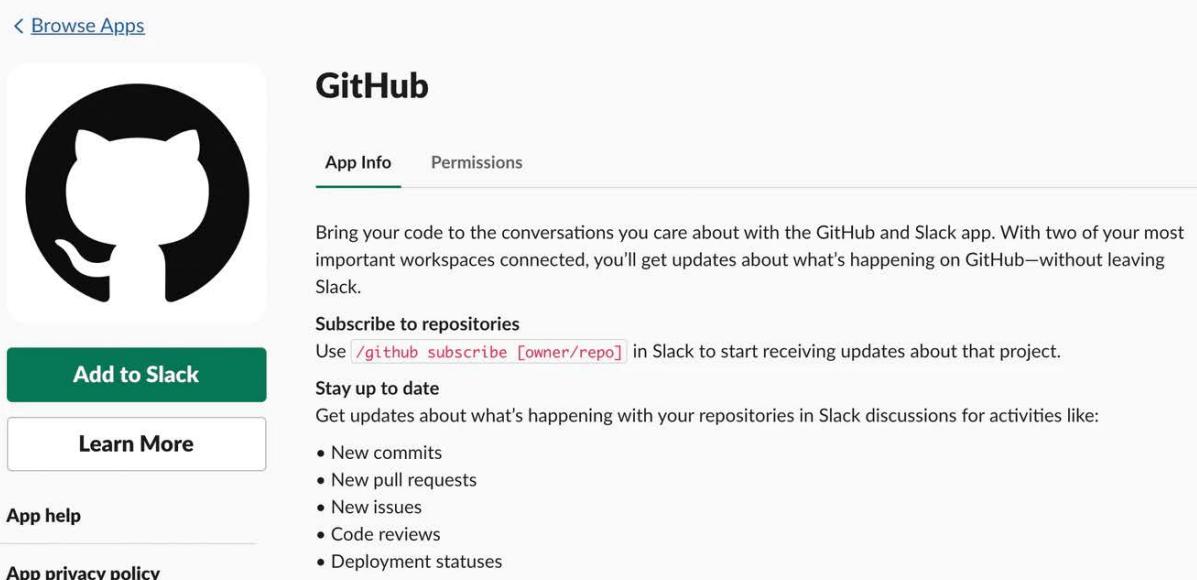
図表9：アクセス権設定（両立させるための新たな統制 1、2、4）

場所・環境等	開発担当者	開発リーダー	役員A	ツール	その他
ワークスペースクラウド (Slack)	メンバー	ユーザー管理者	メンバー	AWS：一部の変更を通知 CircleCI：設定変更を通知 GitHub：設定変更を通知	総務部：オーナー
IaaS クラウド (AWS)	アクセス管理 (IAM) 特権	—	ID新規作成、 権限変更、ID削除 (IAM Administrator Access ポリシー)	—	Terraform：ID新規作成、 権限変更、ID削除 CircleCI：権限変更
	Prd	参照		—	—
	Stg			—	Terraform：リソース新規作成、 変更、削除
	Dev	リソース新規作成、変更、削除		—	—
CI/CD クラウド (CircleCI)	アクセス 管理特権	—	ID新規作成、 権限変更、ID削除	—	—
	パイプライン 設定	参照	新規作成、変更、 削除	—	—
CM クラウド (GitHub)	Main	Write	Admin	—	—
	Release			—	—
	Develop			—	—
	Feature/abc			—	—

図表10：GitHubブランチ保護設定（両立させるための新たな統制 1)¹¹

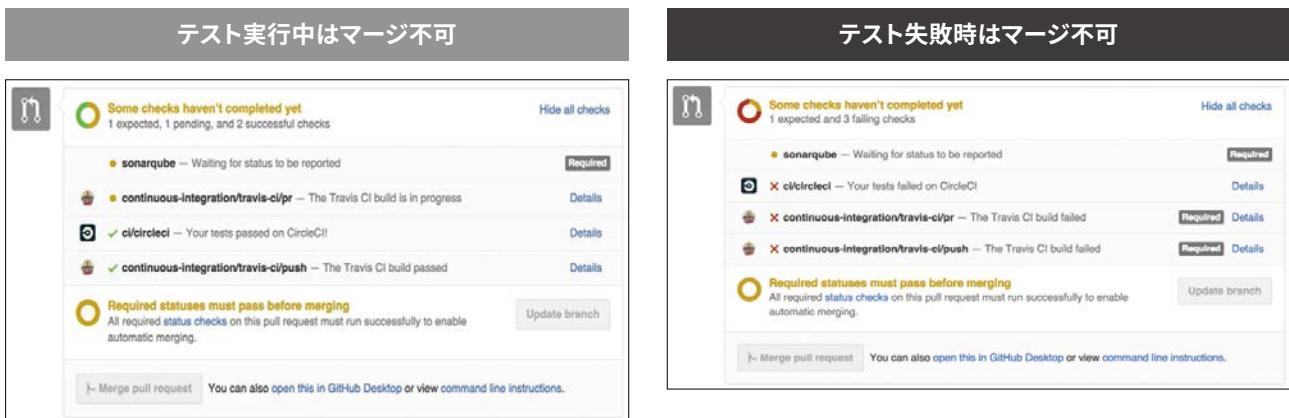


図表11：GitHub通知設定（両立させるための新たな統制 3)

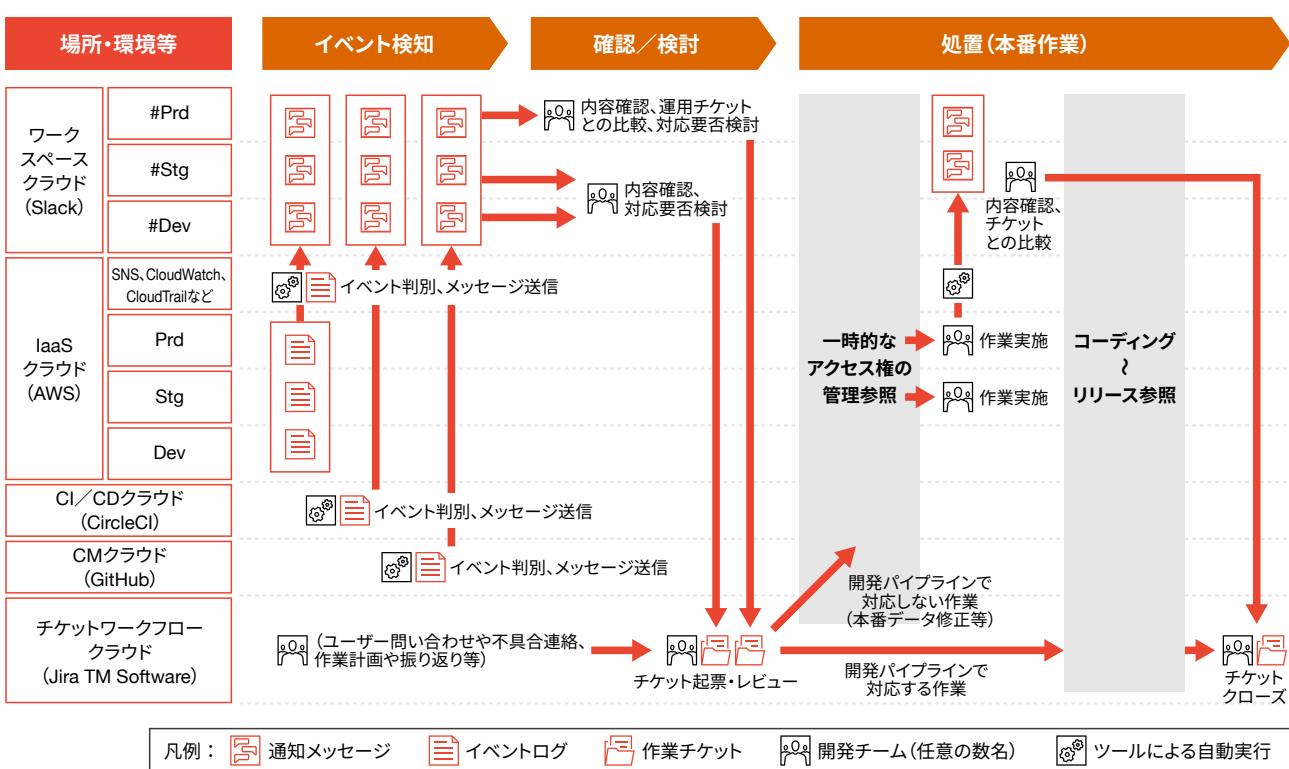


11：ソースコード管理ツールの1つである「GitHub」を利用した場合のブランチ保護の設定画面例。詳細は[<https://docs.github.com/ja/repositories/configuring-branches-and-merges-in-your-repository/defining-the-mergeability-of-pull-requests/managing-a-branch-protection-rule>]を参照のこと。

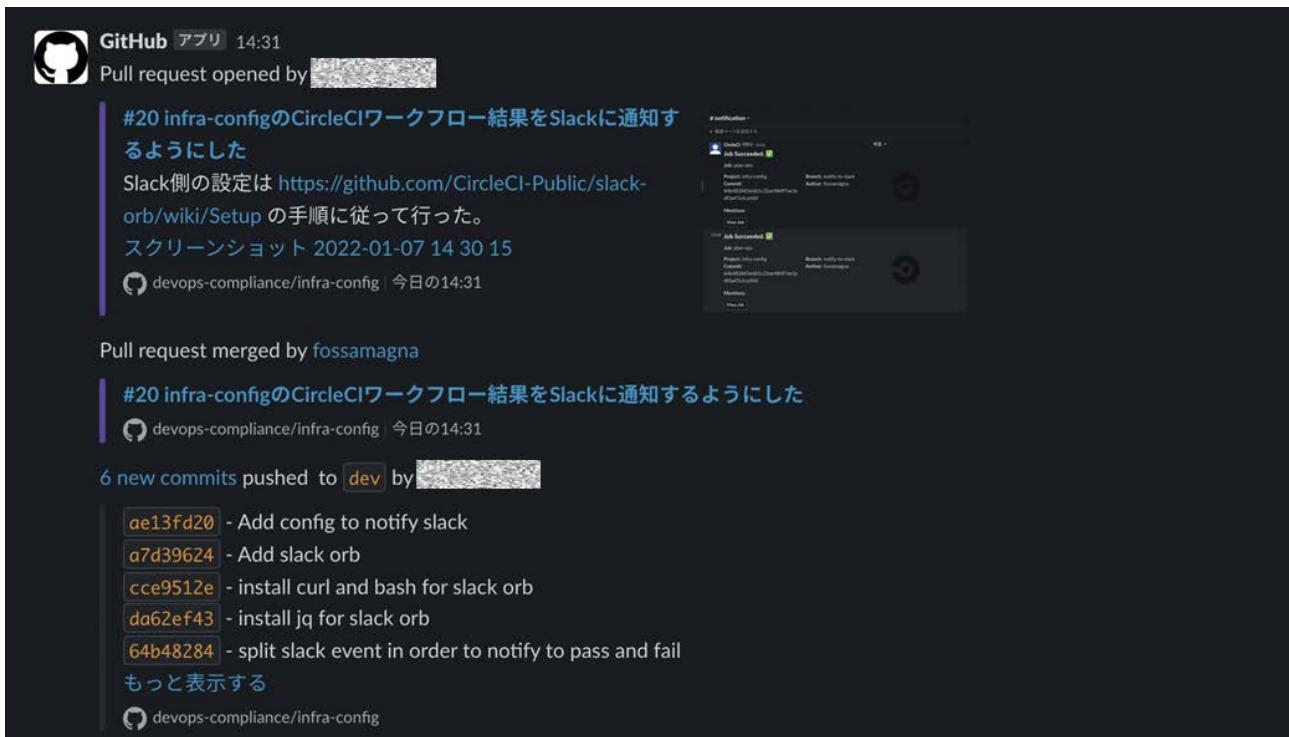
図表12：GitHubによるマージボタンのWarning（両立させるための新たな統制 1）



図表13：システム監視、本番作業（両立させるための新たな統制 3）



図表14：Slackへの通知・確認コメント（両立させるための新たな統制3）¹²



なお、プロダクトオーナー（管掌役員A）は、当該リスクシナリオを重く受け止め、プロダクトZ部門以外への手当てを経営陣と検討しました。

- ・賞与額とローン成約額の連動の廃止
(人事評価制度の見直し)
- ・審査部門による信用スコアの妥当性チェック
(牽制機能の強化) など

12：チャットツールの1つである「Slack」を利用した場合の、CI／CDツール「CircleCI」からの通知メッセージ画面例。画面は私たちで作成。

要件#2 テストと承認への対応

データ、アプリケーションやインフラへの変更が意図したとおり行われるよう、適切なテストと承認を行う

現在の状態

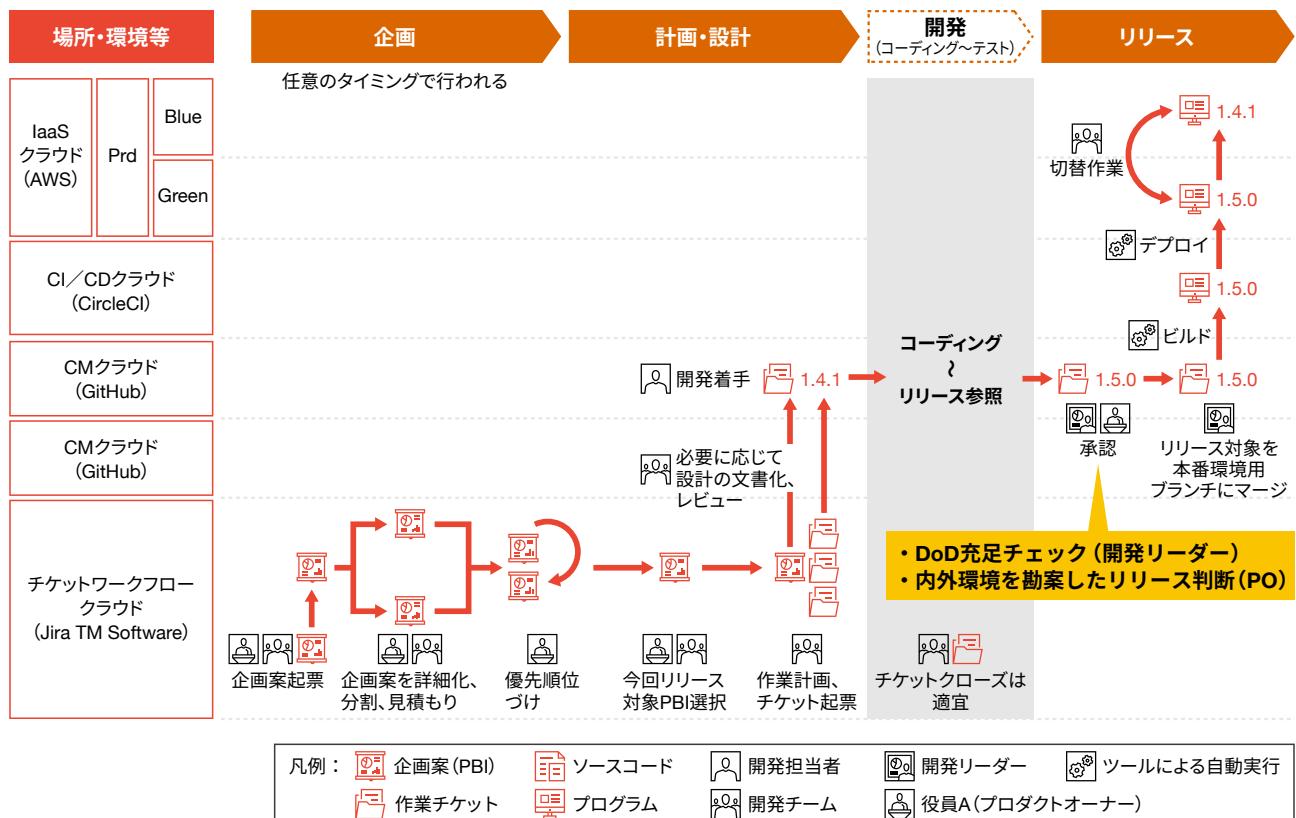
プロダクトZのリリース頻度は2週間に1回です。プロダクトZの変更を伴う企画案について管掌役員A（プロダクトオーナー）を含む部員全員が検討し、「次回何を作るか」はAを含む部員全員が意見を出し合い決定します。

作業の計画（「どう作るか」）や実行は、管掌役員Aを除く開発メンバーに委ねられています。必要に応じて設計文書を作成・更新しますが、全てのリリース対象について作成するわけではなく、コーディングをすぐに行う場合もあります。

開発ではテスト駆動開発を採用しています。開発担当者がプログラム本体のソースコードとテストコードをセットでコーディングし、構成管理ツールに登録することで、CI/CDツールが自動テストを実行します。テスト結果が全て通った（オールグリーン）対象のみ、開発の担当者がレビュー依頼（プルリクエスト）を構成管理ツール上で行います。担当者以外の開発メンバーは、ソースコードとテストコードの内容、テスト結果をレビューします。構成管理ツール上で承認すると、変更されたソースコードとテストコードが結合テストに向かってマージされます。

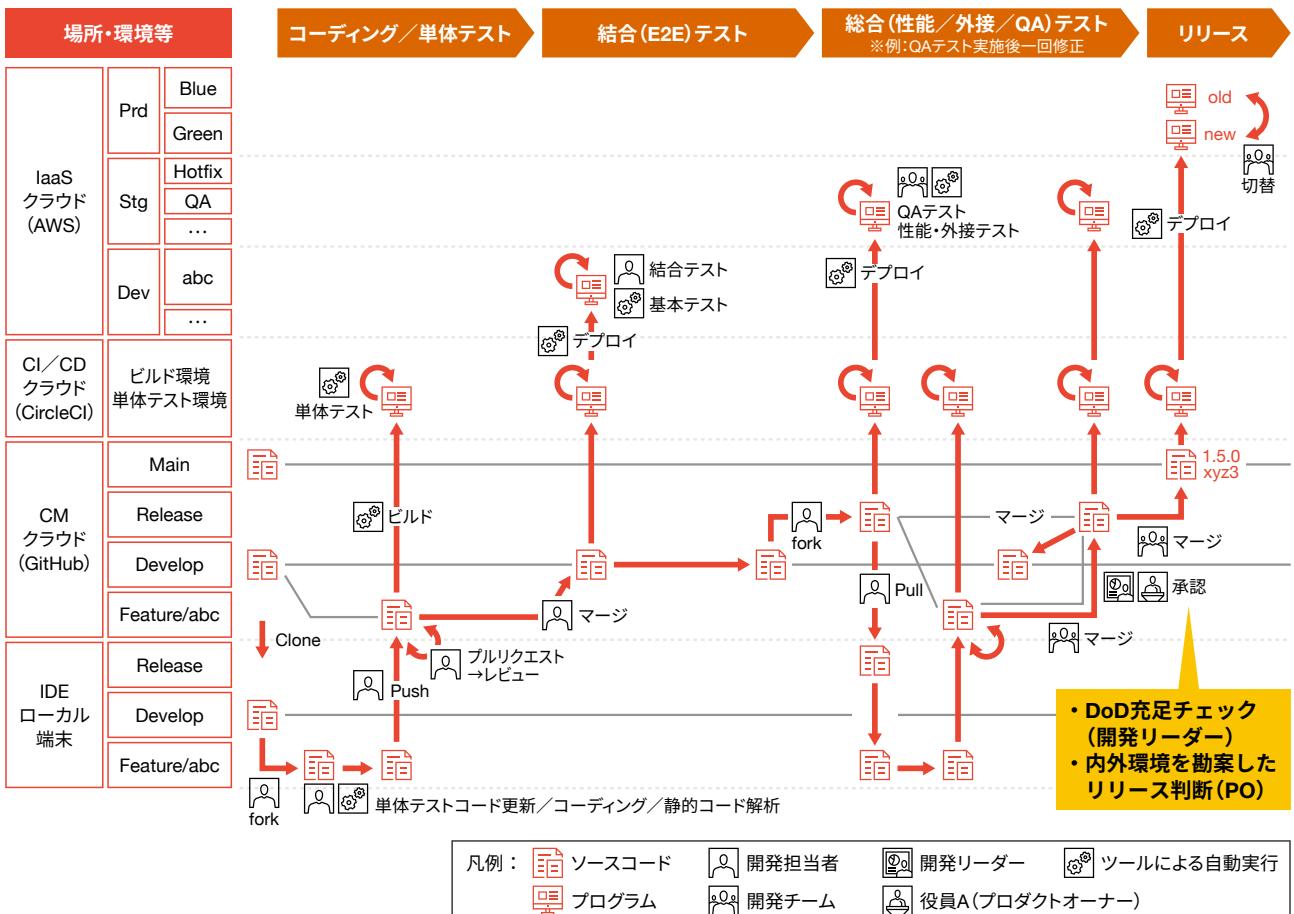
ただし、CI/CDツールの設定上、自動テストなしにプルリクエストを実施することも可能で、しばしばそのようなオペレーションミスが発生しているのが現状です。

図表15：企画～リリース



13: システム開発におけるバージョン管理において、更新履歴を付与していくまとめ。複数名が複数案件による変更を加える際には、このまとめ（ブランチ）をさまざまな単位で分けることがある。このケースでは、開発ステージ（コーディング／単体テスト、結合テスト、総合テスト、リリース）でブランチが分かれている。

図表16：コーディング～リリース



原理・原則とのギャップ

開発対象決定やリリース判断には管掌役員Aが関与しており、伝統的解釈の「上長による承認行為」に相当します。一方で、開発作業においては自動テスト結果を含む部員同士のピアレビューを各ブランチへのマージ時に実行され、承認行為の実行者が上長であるとは限りません。しかしながら、企画の起案とリリース判断のみならず、開発作業における承認の権限までを上長に集中させることは、開発スピードやチーム内コラボレーションを損ないます。そこで、プロダクトZ部門は、原則遵守するための糸口として、まずリスクシナリオを想定しました。

こうした事案はレビュー能力を有した上長の確認、承認行為があれば防ぐことができたと思われますが、プロダクトZ部門は次のように考えました。

- リリース対象選定やリリース判断に、プロダクトオーナーの確認・承認が必要であることは理解できる一方で、中間作業である開発において、その確認・承認を行う者が必ずしも上長である必要はない
- 上述のリスクシナリオへの対応を考えると、中間作業を「上長が承認する」ことは手段の1つである。「意図を実現する変更が加えられている」とこと「余計な変更が加えられていない」ことを、「レビュー能力を有する者」が「必ずレビューをする」ことこそが重要である

リスクシナリオ

プロダクトZ部門には、毎月のように新たなメンバーが参入している。新規参入者には、慣れるまでの2ヶ月間、既存メンバーが寄り添って（バディとして）日々の作業を行うこととしている。

今回、不幸なことに新規参入者Bのバディが入院してしまった。他の既存メンバーで代替することになったものの、どうやら手が回っておらず、ほとんど新規参入者Bの相手ができていない。一方、新規参入者自身Bは自信家であり、「大丈夫です」と淡々と開発・テストを行っていた。

結果、今回のリリースにおいては、新規参入者Bが開発・テストを行った部分でデグレードが発生し、切り戻すことになってしまった。後続の会計システム上のデータへの影響が大きく、決算前に発覚・修正できたものの、財務諸表に重要な虚偽表示が発生し得る事案であった。今回不具合が混入した会計システムへのインターフェース部分は重要機能と認識されており、Dev環境でのリグレッションテストが毎回自動実行されていたが、リリースではそのテストスクリプトが空回り（適切なスクリプトの指定が漏れていた）しており、不具合を検出できなかった。

両立させるための新たな統制

プロダクトZ部門は「要件#2」に対応するため、以下を実装します。

1.自動リグレッションテストのパイプラインへの組み込み

CI/CDツールの設定においては、Devブランチへのマージをトリガーとして、Dev環境へのデプロイと自動テストツールによるリグレッションテストの自動実行を行う。

2.リグレッションテストのテストスクリプトを構成管理対象化

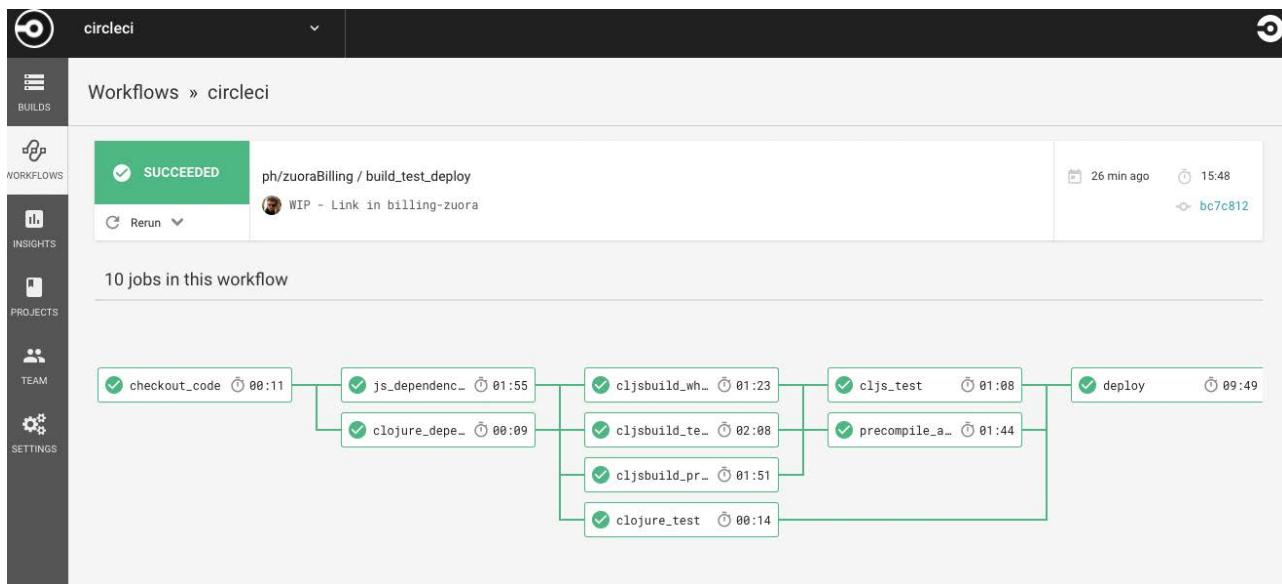
構成管理ツール上で当該テストスクリプトを管理する（ブルリクエストの対象とする）ことで、正式な変更プロセスに則り変更内容に対してピアレビューが実施される状態とする。

3.リリース判定時に各ツール設定の妥当性確認を追加

リリース判定用に用いる完了の定義（DoD）に、「ブランチ保護や自動テスト実行などの各ツール設定が変更されていないこと」「変更されている場合には、その妥当性が複数名により確認されていること」を追加し、テストやレビューが確かに実行されたもののみリリースされる状態とする。

これにより、リグレッションテストの実行を含む、開発中のテストやレビュー行為が、能力を有する者により確実に行われる状態としました。さらに、リリース判定において、管掌役員Aが開発中のテストやレビュー行為が行われているかどうかを確かめることで、開発作業の「承認行為」の要件もクリアしました。

図表17：CircleCIのワークフロー（両立させるための新たな統制 1）¹⁴



14 : CI/CDツールの1つである「CircleCI」を利用した場合の、ワークフロー設定画面例。詳細は<https://circleci.com/docs/ja/2.0/concepts/>を参照のこと。

要件#3 アクセス権を必要最小限に保つ対応

データに対する直接的で不適切な変更ができないように、インフラへのアクセス権を必要最小限に保つ

現在の状態

本番データの修正など、パブリッククラウド上の本番環境への直接アクセスは「必要な時に必要な分だけ権限を付与する」ことが、プロダクトZ部門内で周知されています。本番作業内容をチケットワークフローに起票し、申請者以外の部員が内容をレビューした上で、作業者のIDに権限を付与する運用となっています。一方で、強い権限が付与されたままのIDが長期間残っているケースが散見されていました。また、権限付与の方法は、クラウドが提供する管理コンソールやコマンドラインから開発リーダーが付与する場合もあれば、CI/CDワークフローからスクリプトを実行して付与する場合もあり、明確な手順が決まっていませんでした。

本番データ修正作業は顧客からの問い合わせやクレームから発生することがほとんどであり、その対応速度が顧客満足度に直結するという理解がプロダクトZ部門にはあります。

原理・原則とのギャップ

「本番環境へのアクセス権は必要な時に必要な分だけ付与する」という当初からの方針は、「必要最小限に保つ」という原理原則と整合しています。一方で、実際には強い権限が不要に長期間付与されたままの状態が発生することは問題でした。プロダクトZ部門は、ビジネスパフォーマンスを維持しながら、いかにこの状態を回避するかを、リスクシナリオを想起した上で、予防的な側面と発見・修正の側面から検討しました。

リスクシナリオ

すでに退職が決まっていたプロダクトZ部門のメンバー Cは、障害対応として本番データ修正を行った。データ修正は作業申請のとおりに終えたが、その後の退職日当日に、別の本番データ修正も併せて必要であることに気が付いた。A社に迷惑をかけたくないという思いから、メンバー Cは申請ルールを無視して、まだ剥奪されていなかった本番環境へのアクセス権限を用いて該当の追加的なデータ修正を実行した。事後的にプロダクトZ部門のメンバーへ報告したところ、当該データ修正は不要であることが判明し、データを戻すための再作業が発生した。

両立させるための新たな統制

プロダクトZ部門は「要件#3」に対応するため、以下を実装します。

1. 権限付与方法を「スクリプトの実行」のみとし、コードレビューを必須とする

クラウドの管理コンソールやコマンドラインからの権限付与を不能とし、構成管理ツール上でレビューが通ったスクリプトをCI/CDワークフローが実行することで、権限が付与される仕組みとする。

2. 付与する権限は時限設定（権限失効までの時間設定）を必須とする

権限付与用スクリプトのレビュー時に、時限設定で権限を付与することと、その時間・内容が作業チケットと照らして妥当であることを確認する。

3. 本番作業を重要な行為として通知し、確認する

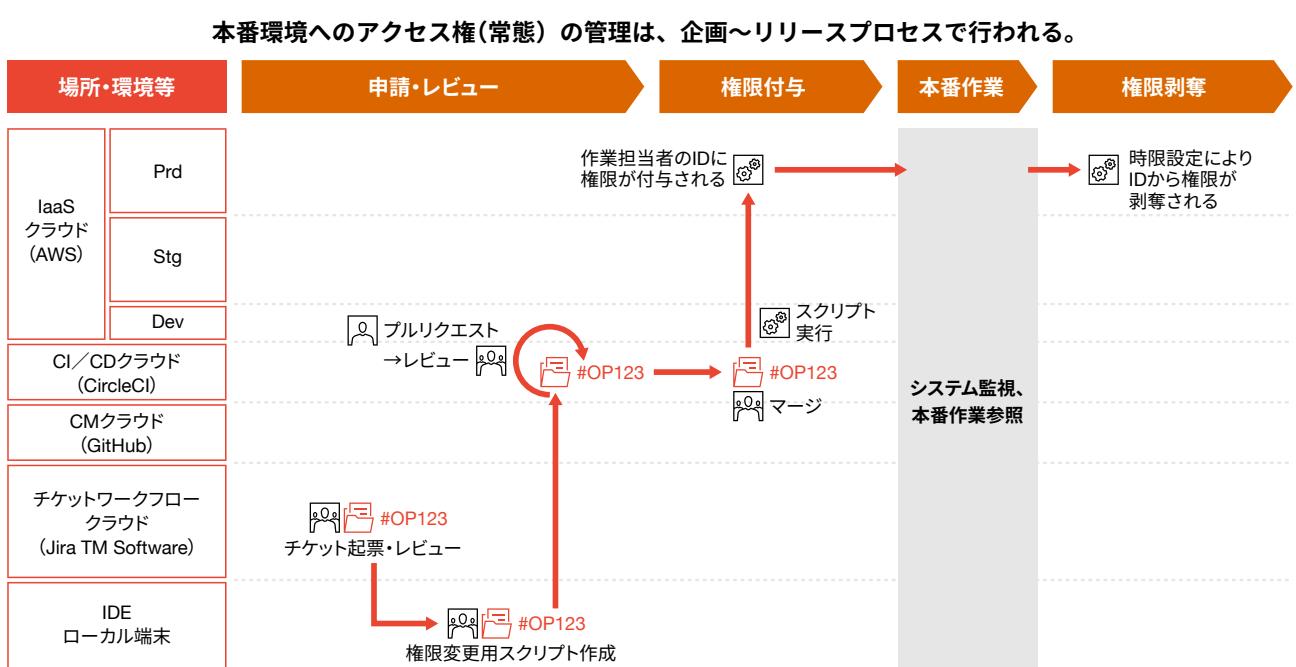
（要件#1と同様）社内チャットツールに監視用チャンネルを作成し、本番環境へのスクリプト実行について通知が挙がる設定とする。挙がった通知については、日次のスタンダップミーティングで、妥当であることを2名以上が確認し、コメントを残すこととする。申請漏れ等の疑義がある場合には、チケットを起票し、開発リーダーの承認を得る。

4. 本番作業時以外（普段の状態）の権限設定の棚卸しを自動化する

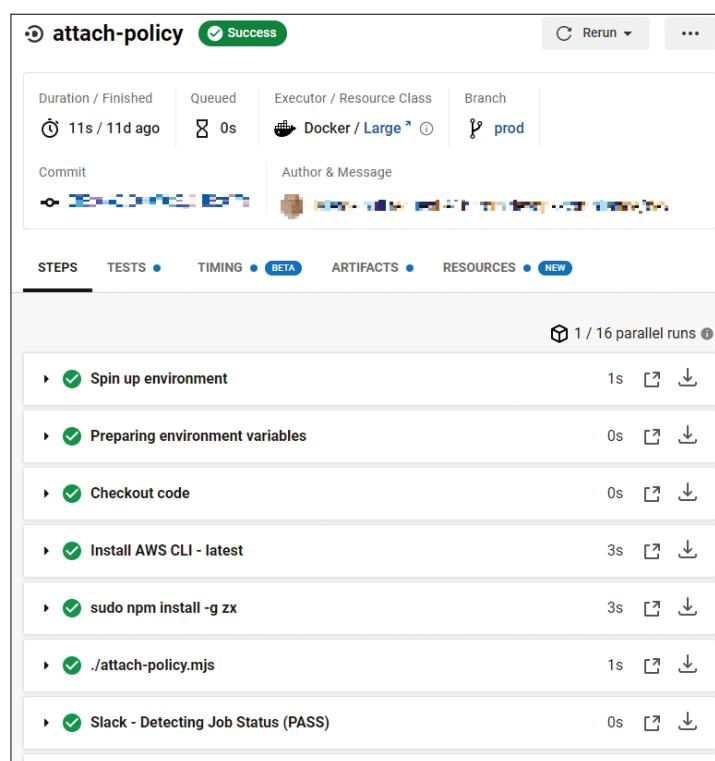
IAM IaCとツールチェーンの権限設定と、会社の人事データベース（協力会社のデータも含む）と紐づく社内インターネットへのログインID権限設定を、ツールによって日次で突き合わせる。結果をチャットツール上の監視用チャンネルに吐き出すこととし、挙がった差異については、日次のスタンダップミーティングで、妥当であることを2名以上が確認し、コメントを残すこととする。

これにより、「要件#3」の実装を確かなものにすることができました。

図表18：本番環境への一時的なアクセス権付与のプロセス（両立させるための新たな統制 1、2）



図表19：CI/CDツールからスクリプトを実行（両立させるための新たな統制 1、2）¹⁵



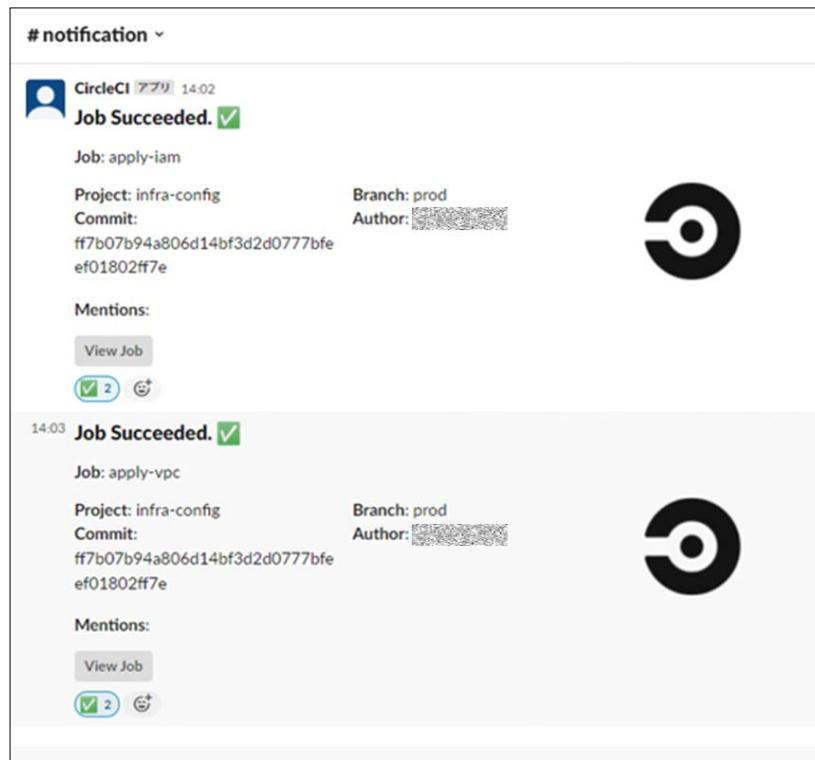
15：権限付与の自動化設定と実行例。CI/CDツールの1つ「CircleCI」を利用し私たちで作成。

図表20：権限付与で使用する定義ファイルの例（両立させるための新たな統制 1、2)¹⁶

```
{  
  "UserName": "ops-user-1", // ポリシーを適用するIAMユーザー  
  "PolicyName": "TemporaryProvieredPolicy", // ポリシー名  
  "PolicyDocument": { // 付与するポリシードキュメント  
    "Version": "2012-10-17",  
    "Statement": [  
      {  
        "Effect": "Allow",  
        "NotAction": [  
          "iam:*",  
          "organizations:*",  
          "account:*"  
        ],  
        "Resource": "*",  
        "Condition": {  
          "DateGreaterThan": {  
            "aws:CurrentTime": "2021-12-10T08:30:00Z" // ポリシーを適用を開始する日時  
          },  
          "DateLessThan": {  
            "aws:CurrentTime": "2021-12-10T15:30:00Z" // ポリシーを適用の期限となる日時  
          }  
        }  
      },  
      {  
        "Sid": "TemporaryProvieredPolicy",  
        "Effect": "Allow",  
        "Action": [  
          "iam:CreateServiceLinkedRole",  
          "iam:DeleteServiceLinkedRole",  
          "iam>ListRoles",  
          "organizations:DescribeOrganization",  
          "account>ListRegions"  
        ],  
        "Resource": "*",  
        "Condition": {  
          "DateGreaterThan": {  
            "aws:CurrentTime": "2021-12-10T08:30:00Z" // ポリシーを適用を開始する日時  
          },  
          "DateLessThan": {  
            "aws:CurrentTime": "2021-12-10T15:30:00Z" // ポリシーを適用の期限となる日時  
          }  
        }  
      }  
    ]  
  }  
}
```

16：サンプルコードは、インフラ環境構築向けコード管理ツールの1つ「Terraform」を利用し、私たちで作成。

図表21：Slack通知サンプル（両立させるための新たな統制3)¹⁷



17：チャットツール「Slack」を利用した場合の、CI／CDツール「CircleCI」からの通知例。画面サンプルは私たちで作成。

推奨事項：可監査性の確保への対応

可監査性の確保：「ルールどおりに運用されている」ことを効率的に説明するために、関連するログの記録と保全を行う

ログの記録と保全の重要性

プロダクトZは、今回のIT監査対応において、親会社内部監査部門から大量のログやチケットデータの提出を求められ、大変な労力を費やしました。「IT統制に対する推奨事項」(P10参照)は、その上で挙げられたコメントです。

本来の監査等では、これ以上に多大な労力を要することを想定されます。「ルールどおりに運用されていること」を説明するために必要な証跡・ログを特定し、取得・保管することとしました。

対象となるログ

全てではないものの、特に重要度が高いと考えられるログは以下(図表22)と考えました。

その内、開発パイプライン(CI/CDツールや構成管理ツール)の動作履歴は大量であり、システム自体の設定変更のログはこれまで全く記録していませんでした。そのため多くのことを考慮する必要が出てきますが、プロダクトZ部門の開発・運用のプロセスは、要求されるIT統制の一部について「自動化されている」「そもそもツールチェーン設計上、リスクがない」ことを説明する上で必要な事項であると判断しました。

この対応により、可監査性を確保することができました。

図表22：可監査性の確保に必要となるログ

開発・運用のプロセス	ツール	必要なログ
企画	チケットワークフロー	<ul style="list-style-type: none">・PBI選定の承認履歴・コーディング計画のレビュー・承認履歴
コーディングおよびリリース	構成管理	<ul style="list-style-type: none">・プルリクエスト、レビュー、マージ履歴<ul style="list-style-type: none">-チケットとのリンクエージ情報を持む-リリース判定時の開発リーダー、POのコメント含む
テスト	CI/CD	<ul style="list-style-type: none">・CI/CDパイプラインの実行履歴<ul style="list-style-type: none">-自動テストの動作履歴含む
本番環境運用	CI/CD	<ul style="list-style-type: none">・ツールチェーン(パイプライン)設定の変更履歴<ul style="list-style-type: none">-特別な権限を有するIDの使用履歴含む
本番環境運用	クラウド上のユーティリティ	<ul style="list-style-type: none">・クラウド上の本番環境の変更履歴・特定IDへの権限付与と時限剥奪の履歴
本番環境運用	チャットツール	<ul style="list-style-type: none">・チャットツール設定の変更履歴

図表23：チケットワークフローに関するAuditログ¹⁸

System						Search Jira admin
Audit Log						Actions
Contains text		Time: All				Export
1-100 of 32220						
Date	Author	Event category	Change summary	Changed object	Actions	
23/Aug/19 1:16 PM	👤 Matt	fields	Custom field created	Team	Show more	
23/Aug/19 1:14 PM	👤 Matt	fields	Custom field created	PC-Ticket	Show more	
23/Aug/19 1:13 PM	👤 Matt	fields	Custom field updated	OKR URL	Show more	
23/Aug/19 12:06 PM	👤 Kevin	workflows	Workflow created	Project BATD: Software Workflow for 22458	Show more	
23/Aug/19 12:06 PM	👤 Kevin	workflows	Workflow created	Project BATD: Software Workflow for 22457	Show more	

18：チケット管理ツールの1つである「Jira TM Software」が提供する監査ログ出力画面例。

詳細は<https://support.atlassian.com/jira-cloud-administration/docs/audit-activities-in-jira-applications/>を参照のこと。





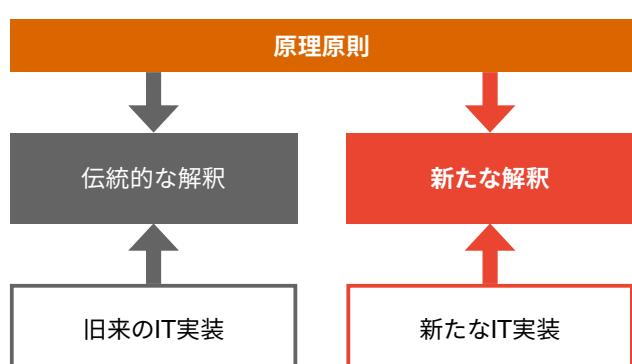
解釈を再鑄造する

ここまで見てきた、開発と運用の分離、中間成果物の上長承認、本番アクセス権の限定は、DevOpsと衝突を起こしやすい典型的な伝統的解釈です。この衝突に直面したプロダクトZ部門は、原理原則の遵守を伝統的解釈ではない方法で達成するべく、自部門の環境を整理し、リスクを想定し、本質的な対応を検討しました。その結果、原理原則を遵守することが可能な解釈を作り上げることができました。

また、プロダクトZ部門の対応は、原理原則の遵守を達成したというだけでなく、副次的な効果も発揮しています。具体的には、今回の対応は開発・運用時のコーディングエラー、デプロイの誤り、レビュー漏れ、本番環境のデータ保護といった、プロダクト品質にも寄与する内部統制の構築につながっています。これは、原理原則に立ち返って、本質的な検討を実施した効果に他なりません。

ITの統制に対する伝統的な解釈を見直し、原理原則に立ち返ってDevOpsの環境のリスクを検討し、コントロールを構築する。それにより、コンプライアンス要件とDevOpsの関係性を、衝突から両立可能なものへと変化させることができます。その結果、DevOpsの信頼性を高めながらビジネスパフォーマンスを向上させていくことができるのではないかと、私たちは考えます。

図表24：新たな解釈を導出する





おわりに

本レポートではITコンプライアンス対応に悩むDevOps実践組織に対して、「解釈を再铸造する」という解决の方向性を示しました。とはいえこれは簡単なことではありません。

長い期間をかけて形成された解釈には一定の根拠があり、ある場面では伝統的解釈は機能するものですし、一方で、新たな場面において再铸造した解釈が機能するかどうかは、将来におけるリスク対応力とビジネスパフォーマンスによってのみ確かめられるからです。

新しい実装が高速で進む現在において必要なのは、紋切り型のコンプライアンス対応や指摘ではなく、未来を見据えた対話に基づく本質的な対応です。プロダクトが社会の一部として機能する以上、原理原則を明らかにし、本質的な対応を行っていく姿勢が、関係者全員に必要なのではないでしょうか。

本レポートはGitHub上で広くご意見・ご感想を募集する予定です。ぜひ、Issueチケットを通して対話をまいりましょう。

プロダクトチーム¹⁹の皆様へ

「監査対応」が発生しているあなたのビジネスは、いわば順調に拡大しているということです。それは心が折れる作業かもしれません、コンプライアンス要件には、あなたの働き方に影響する内容が含まれています。画一的な対応はあたかもボディブローのようにビジネスに影響を与えるでしょう。もし、あなたがあなたのプロダクトについて、ビジネス上のコミットメントを持っているのであれば、コンプライアン

ス対応もその一部です。まるでPBIを起票するように（あるいは実際に起票して）タイムリーに、真正面から対応すべき内容です。そうすることで獲得した「私たち流の」働き方によって、より高い品質を確保し、市場シェアを伸ばし、取引先を増やし、競合の一歩先へとビジネスを進めることができるでしょう。原理原則とその背景（歴史的事案や社会的動向）を内部監査人から学ぶことは、その第一歩となります。

内部監査人²⁰の皆様へ

本レポートは、これまでの実務を否定するものではありません。アシュアランス業務はそれ自体がリスクの高い業務です。一步間違えば、アシュアランス提供側の存続が危うい事態になります。確かに、本レポートが提唱するような新たな解釈を交えながらアシュアランス業務を行うことには、多大な労力がかかります。しかし、被監査主体のパフォーマンスに配慮できるアシュアランスと、そうでない紋切型のアシュアランスのどちらがビジネスに貢献できるかを勘案すると、この労力はかける価値があります。

手元のチェックリストの更新は、アシュアランス提供前に着手可能であり、事前に準備しておくことで、アシュアランス提供時のリスクを十分に低減できます。今一度、手元のチェックリストを確認しましょう。全てがそうではありませんが、たいていの場合、「本書に記述された項目は画一的に適用されるべきものではない」「内外環境を勘案してリスクベースで適用するものである」に類する記述があるのではないかでしょうか。さあ、今すぐ手元のチェックリストを更新しましょう。原理原則の背景と、新たなIT実装を踏まえながら。

経営者の皆様へ

世間でデジタルトランスフォーメーションが声高に叫ばれるようになり、あらゆる事業体がデジタルを取り入れています。社会がSociety 5.0²¹化するのは、そう遠くない未来です。このようなビジネス環境において、あなたの組織のITとコンプライアンスを調和させることは、「トラスト」を得ていくための重要な論点です。

「トラスト」とは、あるリスクが妥当な水準にコントロールされていること、そして、そのことに対する外面的印象を指します。各組織がサイバー空間内で細かい単位でつながるこれから社会において、重要な概念です。そして、本レポートでA社が実施した「再铸造」は、狭義のコンプライアンス対応ではなく、まさにこの「トラスト」の醸成だと言えます。A社が実施した

のは「能動的なcomply」と「積極的なexplain」です。本来のコンプライアンス原則に立ち返ってリスクコントロール実装の検討を能動的に行い、指摘した側と積極的に対話を続けることで、ビジネススピードを損なうことなく、DevOpsとコンプライアンス要件を調和させることができました。

監査や認証といったアシュアランスを得ることは、トラスト醸成の主たる手段です。その対応が単なる追加コストで終わるのか、あるいはトラストにつながる投資となるのかは、経営の方向づけそのものではないでしょうか。コンプライアンス要件の原則を見つめ、貴組織だからこそ「再铸造」を行いませんか。

19：本ケースにおけるプロダクトZ部門のような集団。開発者だけでなくプロダクトオーナーを含む。

20：いわゆる第3線が実施する内部監査に加えて、第2線による点検やモニタリングを実施する者も対象となる。

21：内閣府の「第5期科学技術基本計画」（2016年）において、「ICTを最大限に活用し、サイバー空間とフィジカル空間（現実世界）とを融合させた取組により、人々に豊かさをもたらす『超スマート社会』を未来社会の姿として共有し、その実現に向けた一連の取組」と定義されている。

Terraformは、HashiCorp, Inc.の商標または登録商標です。

GitHubは、GitHub, Inc.の商標または登録商標です。

Jira TM SoftwareはAtlassian Pty Ltd. の商標または登録商標です。

Slackは、Slack Technologies, Inc. の商標または登録商標です。

CircleCIおよび製品ロゴは、Circle Internet Services, Inc.の商標または登録商標です。

AWSは、米国その他の諸国における、Amazon.com, Inc.またはその関連会社の商標です。

執筆者

永和システムマネジメント

永和システムマネジメントは、日本において、2000年からアジャイルソフトウェア開発を実践しております。コーチング・教育・エンジニアリングを得意とする専門家チームが、金融・医療・組込みなど幅広い事業分野に対しサービスを提供しています。2018年には、アジャイル開発拠点である Agile Studio を創設し、技術的卓越性によってお客さまと共に成長できる、共創・共育型のビジネスとソフトウェアづくりを推進中です。

Agile Studio は随時リモート無料見学会を実施しており、これまで1,000人以上の方がアジャイルなソフトウェア開発の現場を体験されています。詳しくは、agile-studio.jp をご覧ください。



平鍋 健児
代表取締役社長



岡島 幸男
取締役CTO／
Agile Studio
ディレクター



村上 雅彦
Agile Studio
エンジニア

PwCあらた有限責任監査法人

PwCあらた有限責任監査法人は、PwCグローバルネットワークのメンバーファームとしてデジタル社会に信頼を築くリーディングファームとなることをビジョンとしています。世界で長年にわたる監査実績を持つPwCネットワークの監査手法と最新技術により世界水準の高品質な監査業務を提供するとともに、その知見を活用した会計、内部統制、ガバナンス、サイバーセキュリティ、規制対応、デジタル化対応、株式公開など幅広い分野に関する助言（プローダーアシュアランスサービス）を通じて社会の重要な課題解決を支援しています。



宮村 和谷
パートナー



佐藤 要太郎
シニアマネージャー



伊藤 英毅
シニアマネージャー



横山 和典
マネージャー



小田切 洋介
シニアアソシエイト

お問い合わせ先

PwC Japanグループ
<https://www.pwc.com/jp/ja/contact.html>



永和システムマネジメント Agile Studio
<https://www.agile-studio.jp/>



www.pwc.com/jp

PwC Japanグループは、日本におけるPwCグローバルネットワークのメンバーファームおよびそれらの関連会社（PwCあらた有限責任監査法人、PwC京都監査法人、PwCコンサルティング合同会社、PwCアドバイザリー合同会社、PwC税理士法人、PwC弁護士法人を含む）の総称です。各法人は独立した別法人として事業を行っています。

複雑化・多様化する企業の経営課題に対し、PwC Japanグループでは、監査およびアシュアランス、コンサルティング、ディールアドバイザリー、税務、そして法務における卓越した専門性を結集し、それらを有機的に協働させる体制を整えています。また、公認会計士、税理士、弁護士、その他専門スタッフ約9,000人を擁するプロフェッショナル・サービス・ネットワークとして、クライアントニーズにより的確に対応したサービスの提供に努めています。

PwCは、社会における信頼を築き、重要な課題を解決することをPurpose（存在意義）としています。私たちは、世界155カ国に及ぶグローバルネットワークに284,000人以上のスタッフを擁し、高品質な監査、税務、アドバイザリーサービスを提供しています。詳細はwww.pwc.comをご覧ください。

発刊年月：2022年3月 管理番号：I202107-09

©2022 PwC. All rights reserved.

PwC refers to the PwC network member firms and/or their specified subsidiaries in Japan, and may sometimes refer to the PwC network. Each of such firms and subsidiaries is a separate legal entity. Please see www.pwc.com/structure for further details.

This content is for general information purposes only, and should not be used as a substitute for consultation with professional advisors.