Infrastructure as CodeとServer Lifecycle Automationの活用

コード管理によるITインフラ運用自動化の実践

ITインフラ運用効率化のためには優先度を決めて運用自動化に取り組む必要があります。コードを使ってシステムを管理することで運用の効率性や正確性が向上し、作業ミスの削減や必要な時に即座にシステムを復元できるようになります。「Infrastructure as Code」により複雑な設定項目を持つシステムでも自動化運用が実現でき、「Server Lifecycle Automation」によりカタログ化されたツールによるインフラの自動化運用が可能になります。

1. はじめに

企業IT動向調査2017[1]によると、「ITインフラにおける企業の優先課題」の1〜3位が、前年度調査に引き続き、「セキュリティー」「運用コスト削減」「運用省力化」となっています。こうした企業のインフラ運用効率化が大きな課題となっていることが分かります。企業におけるインフラ運用には、クラウドの普及に伴い、「ビジネス要求に応えるための迅速な対応」と「強固で堅実なセキュリティーへの対応」が同時に求められています。スピードと安全の両面の要求が折り重なり、インフラ運用に関する作業量が指数級数的に増大し負担となっているのです。

すでに多くの企業でソフトウェア・ロボットを使って業務プロセスを自動化するRPA(Robotic Process Automation)などによるオペレーション自動化やシェル・スクリプトなどによる効率化への取り組みが見受けられます。しかし、運用を効率化するための重点を決めるノウハウが少なく、確固たるソリューションが不足しているため一つの企業内の努力では成果は限定的になりがちです。運用効率化のためには、サーバー構築および運用を含むライフサイクル全般を考慮する必要があります。IBMでは運用自動化を行うための全方位的なソリューションを提供し、豊富な運用実績に基づいたアプローチを提案することで、企業のインフラ運用の効率化を実現していきます。

本稿では、2章で運用自動化のアプローチ、3章でサーバーなどの構成をコードとして扱う管理手法である「Infrastructure as Code」について解説します。4章ではIBMの自動化ソリューション「Server Lifecycle Automation」を説明し、インフラ運用の効率化についてのIBMの取り組みを紹介します。

2. ITインフラ運用自動化のアプローチ

自動化する対象の優先度を決めることは大切なことで

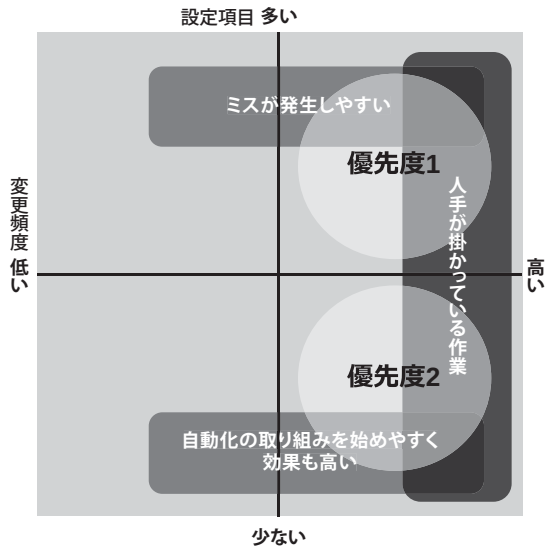

図1. インフラ作業の分類と自動化への取り組みの優先度

す。インフラ運用の現場では多種多様な変更手順や対応すべきITIL準拠プロセスが存在します。コンプライアンスの順守も重大でサーバーなどの設定を常に正しく管理することが求められ、作業が複雑化しています。運用者が作成するスクリプト・プログラムによる自動化も限定的で、多くの人手が掛かっています。インフラの変更作業は設定項目や設定内容の種類が多く特に人手が掛かり、作業ミスも起こりやすくなっています。**図1**に示すとおり、「変更頻度が高く設定の種類が多いシステム」や「変更頻度が高く設定の種類はそれほど多くないシステム」では、人手による作業を自動化していくことで作業の効率化や正確性を上げる効果が見込めるため、それぞれの特性に合わせながら優先して取り組む必要があります。

インフラ全体をコードで管理することで、これまでにない手法で安全で柔軟な運用が実現できます。基本的な設定の変更作業やセキュリティー対応など時間が掛かる繰り返し行われる作業を自動化することは作業の効率性を高めます。また、「同じ環境下で同じコードを同じ回数だけ流せば同じ状態のシステムをもう一度作り出すことができる」という冪等性（べきとうせい）のある運用の実現も運用の即応性・柔軟性を高めます。自動化は同じ作業を繰り返し実行できることも大事ですが、別の環境でコードを実行するだけで同じサーバーを即座に作れる運用を実現することがサイトの継続性を高めます。システム設定を記述したコードをライブラリー管理することで、別のクラウド・データセンターに既存の環境をすぐに移植できるという動的な運用や、システムが壊れたときには作り直せばよいというイミュータブル・インフラストラクチャーの考え方を適用できる運用が可能になります。コードによる管理は、クラウドの単一障害点やサイバー・セキュリティーといったニーズにもつながっています。

日常的に発生するセキュリティー対応などのように変更頻度は高いが設定項目の種類が少ない領域については、IBMが提供するServer Lifecycle Automation（以下、SLA）のようにカタログ化されたツールを活用することで、日常作業の多くを自動化して効果を得ることができます。システム運用の歴史が長く運用が安定していて変更の種類が限定されているようなシステムに対しても、IBMは豊富なユーザー・エクスペリエンスに基づいた数多くの自動化ソリューションを提供しています。

一方、変更頻度が高くより複雑な設定項目を持つ新しいシステムの自動化についてはInfrastructure as Code（以下、IaC）を適用し「Chef」などに代表されるソフトウェアによるサーバー構成ツールを活用します。クラウドの普及によりサーバーの再作成が容易となり構築の柔軟性が高くなっている環境を利用し複雑化した状況下でも、利用者が実現したい自動化運用をカスタマイズして実装できるようになります。

SLAやIaCの手法は、サイバー・セキュリティー・リスクの高いインターネットのシステムや災害やクラウド障害というような避けがたい障害を回避する際にも利用可能です。こうした手法により短時間で変更履歴を再生

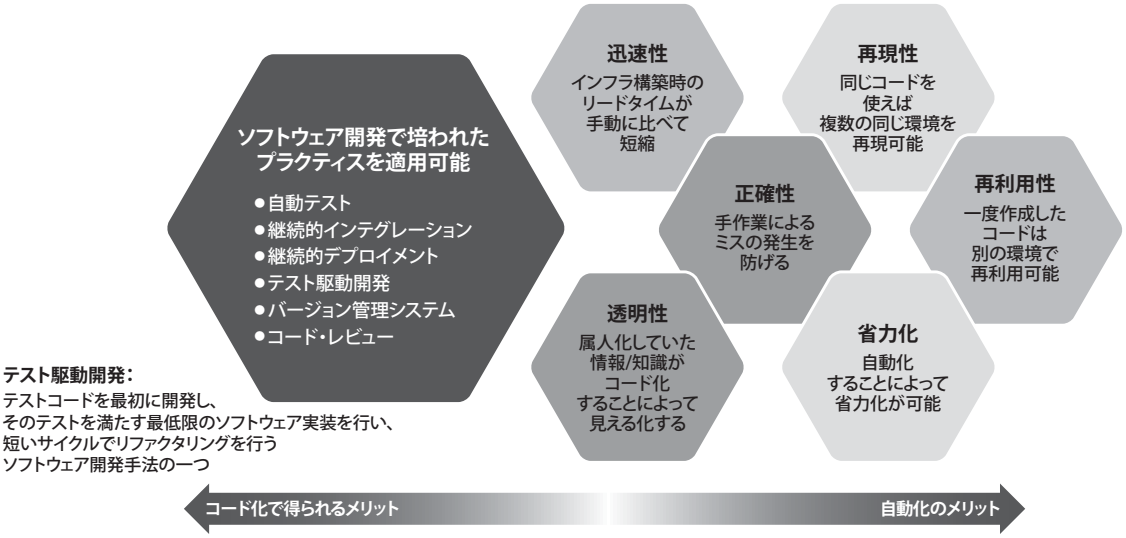

図2. Infrastructure as Codeのメリット

して正しい状態のシステムを再構成することが容易になります。デジタル・ビジネスのような停止が許容されないリスクの高いシステムにおいても、DDoS攻撃やインジェクション・クラウド障害などサイトの停止につながるような緊急事態に対して、サイトの移設とシステムの再作成・自動化運用を再現できるようになります。ここでは、省力化する自動化そのものが目的ではなくインフラ運用の回復性（Site Reliability）を実現することが重要です。

3. Infrastructure as Code（IaC）

3-1. Infrastructure as Code とは

IaCとは、インフラの構成・設定を自動化コードなどの外部ファイルに記述することでインフラの構成・設定を管理する手法です[2]。従来のインフラ構築では設定手順書が存在し手作業でサーバーなどを設定していました。そのため実際の構成はサーバーそのもの、あるいは人間の書いた報告書にしかありませんでした。IaCでは、設計書や設定手順書ではなく自動化ツールの処理を記述したコードを管理します。例えば、MariaDBのインストール・パスワードやパラメーターは後述するサーバー構成ツールが定めた言語で記述されます。そのコードをサーバーに適用することで、コードで記述されたとおりのサーバーが構築できます。クラウドのようにスケールアウトするシステムでは、何百台ものサーバーの構築は膨大な作業量になり手作業によるミスも発生し、テストや確認作業にもまた膨大な作業が発生します。IaCにより大量のサーバーに対してコマンド一つで同じ作業を繰り返し実施できるようになり、作業ミスも減らすことができます。

以下に、IaCによる自動化のメリットをまとめました。

(1)サイバー攻撃やクラウドの重要障害、災害からの退避
(2)インフラ構成のバージョン管理
(3)継続的インテグレーション（CI※1）と継続的デプロイメント（CD※2）によるリリースの高速化

図2に示すとおり、CI/CDはソフトウェア開発のプラクティスをインフラ領域に取り入れるため、IaCはDevOps※3を構成する要素の一つとも言われています[3]。

※1 CI:Continuous Integration、不具合の早期発見を目的に頻繁にビルドを繰り返すソフトウェア開発手法の一つ
※2 CD:Continuous Deployment、完成したコードを自動的に本番環境にデプロイするソフトウェア開発手法の一つ
※3 DevOps:開発担当者と運用担当者が連携して協力するソフトウェア開発手法の一つ

3-2. Infrastructure as Codeのツール

多くの効果が期待できるIaCですが、その実現にはコードを実行する環境をつくるためのツール群が必要です。IaCの適用範囲はインフラのプロビジョニングからサーバー破棄まで多岐にわたるため、環境を領域に分けてそれぞれの領域で適したツールを使うことが一般的です[4]。**表1**には4つの領域ごとに代表的なツールの役割を示しています。

IaCで中心的役割を果たすのはサーバー構成ツールです。サーバー構成ツールの有名なものとして、前述したChefをはじめ「Ansible」「Itamae」などがあります。その中でも対応するプラットフォームの多さやプログラマブルな記述の容易性からIBMが積極的に採用しているのがChefです。多くの時間が費やされているOSやミドルウェアの設定変更やパッチ適用などのサーバー内の構成をコード化できるため、IaCにはサーバー構成ツールから取り組むことを推奨しています。サーバー構成に特化した宣言的なDSL（Domain Specific Language）で書かれたChefの自動化コードはCookbookと呼ばれ、サーバー構成をシンプルに記述できるという特徴を持っています。すべての構築作業をChefで自動化したケースでは、手作業による構築と比べて94%、ハイパーバイ

表1. IaCの環境を構成するツールの分類

分類	説明
ダイナミック・インフラストラクチャー・プラットフォーム	ソフトウェアで定義可能なサーバー、ストレージ、ネットワーク資源を提供するITインフラ。
インフラストラクチャー定義ツール	ITインフラの資源構成や設定をすることができるツール。
サーバー構成ツール	サーバー内の構成や設定を管理する構成ツール。
インフラストラクチャー・サービス	モニタリング、分散処理管理、ソフトウェア・デプロイなどのITインフラ管理を支援するツールやサービス。

ザーのサーバー複製機能を活用した構築と比べても約90%の作業時間削減を実現した実績があります。

3-3. バージョン管理と構成の多様性への対応

エンタープライズITでIaCを実現するにはソフトウェアと同様にコードのバージョン管理が重要です。ここでは、Chefを効率的に活用するためのCookbookの書き分け方について解説します。

小規模システムの場合、Chefで構築する対象のサーバー設定の種類も少ないため、一つのCookbookの中にすべての設定ロジックと設定値をまとめて書くことが多くなります。一方、大規模なITシステムでは扱うサーバー設定の種類が多くなり一つのCookbookの中にすべての設定ロジックと設定値をまとめて記述すると、設定のバリエーションに対応するために分岐が複雑になったり同じ設定ロジックが何度も登場したり、可読性が悪くなる問題がでてきます。Chefの標準機能にはサーバー設定のバリエーションに応じた設定値を付与することができるEnvironmentやRoleの機能がありますが、設定値をChef内部で管理してしまうためgitのようなバージョン管理ができません。

これらの問題の解決策として、IBMの自動化推進チームはWrapper Cookbookモデルを提唱しています。**図3**に示すように、(1) 設定ロジックが書かれたCookbook、(2) ITシステム共通設定が書かれたCookbook、(3) サーバーごとの独自設定が書かれたCookbookという役割ごとのコードを用意し、include機能でCookbookを階層化することで、バージョン管理可能で可読性の高いCookbook群を作ることができます。

4. Server Lifecycle Automation(SLA)

SLAは、特に人手が掛かりミスの発生に繋がりやすい作業に着目したソリューションです。運用自動化により正確性と効率性を上げ、より生産性の高い作業に人を集中させることができます。IBMはこれまでの膨大な運用の経験に基づき作業対象を選択したカタログとグローバルな運用チームによる自動化の仕組みを提供することで、効率的な自動化を提供します。カタログ上で選択できる自動化コンポーネントは、お客様サイトやクラウド上のシステムのいずれに対してもアドオンして構築・運用の標準化を実現することができます。

SLAは、「IBM Services Platform with Watson」(技術解説「IBM Services Platform with Watson」44ページ参照)の運用ツールの一つとして「IBM Business Process Manager」(ビジネス・プロセスを可視化して管理するソフトウェア)や前出のChefを活用し、特に工数が掛かっている構築・運用作業を自動化します。SLAは以下に述べるようなコンプライアンス、パッチ適用、サーバー構築などの自動化コンポーネントを統合したもので、共通のユーザー・インターフェースで、複

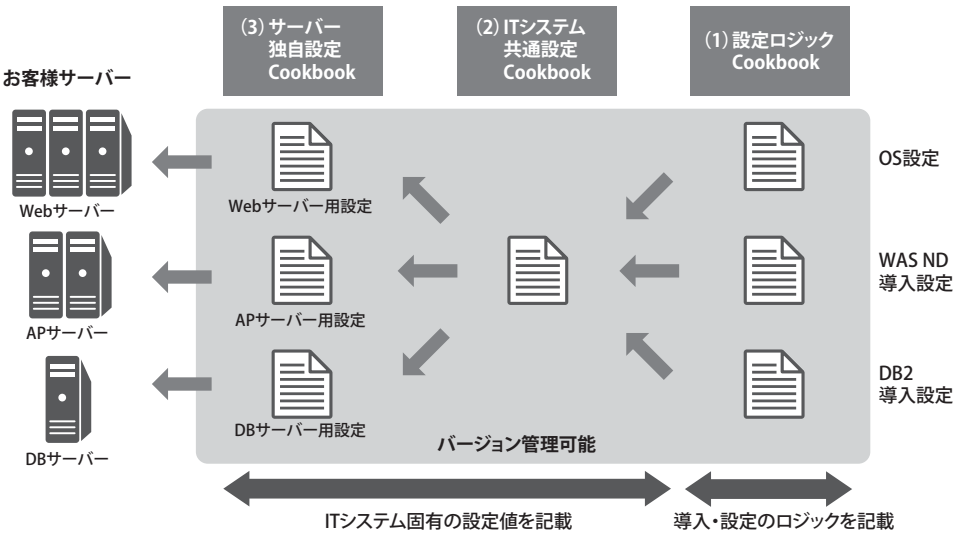

図3. Wrapper Cookbookモデル

雑な操作を伴わずすべてのツールの操作が行える総合的なプラットフォームです（**図4**）。

4-1. Continuous Compliance

システムの健全性を維持するために定期的に実施されるヘルスチェックと、結果として発見された逸脱項目を修正しシステムを健全な状態に保つための修正作業には多くの時間を要しています。手動でのヘルスチェックと修正作業ではミスも避けられず、重大なセキュリティー・リスクを招くこともあります。「Continuous Compliance」（以下、CC）は、こうしたヘルスチェック作業とそれに対する修正作業やその前後で必要となる変更承認などのプロセスを含めて自動化することができ、作業の効率化と正確さを実現します（**図5**）。CCを利用したヘルスチェック運用では基本的に毎日の定期的なヘルスチェックと修正作業を行い、常にシステムを健全な状況に保ちます。例えばソフトウェア導入のために一時的に変更されたシステム・ファイルの権限なども、1日で元の健全な状態に復帰し作業者の戻し忘れなどを防ぎます。また、突然のシステム監査にも慌てることなく対応できるダッシュボードを提供します。

4-2. Risk based Continuous Patch

コンプライアンス同様、セキュリティーの脅威に対応するために適用が必要となるパッチの選択・適用・管理にも多くの時間を費やしています。「Risk based Continuous Patch」は、実機上のパッチ適用状態の収集、パッチ適用の判断、パッチ適用状態の管理、その前後で必要となる承認などのプロセスも含めた自動化を実現しています（**図6**）。パッチ適用作業そのものだけでなく、これまでエクセルやそのほかのツールを使って管理されていたパッチ適用状態の管理も含めて実機上のデータを元に正確に管理できるようになり、作業の効率化と管理の正確性が担保されてシステム監査にも柔軟に対応できる運用が実現できます。

4-3. Self Service Delivery

「Self Service Delivery」（以下、SSD）は、変更作業の自動化を実現します。SSDは、IBMのこれまでの運用ノウハウから多くのお客様で共通に実施される変更作業をカタログで提供しています。Chefなどのサーバー構成ツールと異なり、個別にコードを準備することなくカタログの中から必要な変更作業を選択して実行することが可能です。また、変更作業前後の承認などのプロセスもSSDの中で組み込まれているため、統一的な自動化対応が可能です。SSDは今後、より多くの自動化された変更作業の提供や外部の変更管理ツールとの連携なども予定しています。これにより変更管理から変更作業までより広く統合して自動化することができ、ミスの少ない正確で効率的な運用を実現できます。

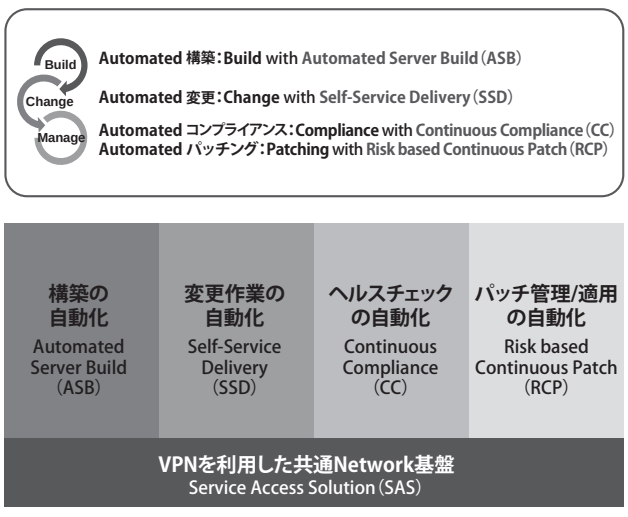

図4. Server Lifecycle Automationが提供する自動化コンポーネント

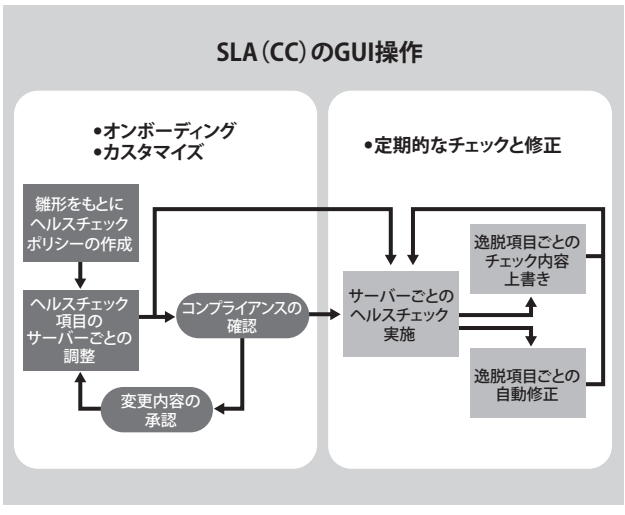

図5. Continuous Complianceによるヘルスチェック

4-4. Automated Server Build

「Automated Server Build」(以下、ASB)は、仮想化された環境を前提としたサーバー構築に関連する作業の自動化を実現します。ASBではサーバー・イメージからのサーバー作成を共通のユーザー・インターフェースから行えます。サーバー作成後に必要となるOSの導入／設定、ツールやミドルウェアなどの導入／設定、その後のヘルスチェック対応やパッチ適用、サービスインに必要となる管理プロセスの実施も含めて、一連の構築作業としてパターン化し繰り返し構築に利用できます(**図7**)。アプリケーション部門の要求に応じてサーバーの払い出しを頻繁に行うプライベート・クラウド環境の運用や、緊急事態に別の環境に同様の構成のサーバーを即時作成するなどの運用に活用することができます。

4-5. SLAを利用するためには

IBMのオートメーション・ツールは、クラウド環境やサービス・プラットフォームの考え方に基づいたサービス提供に対応するためにインターネット接続を前提としています。SLAの接続に用いられる共通のネットワーク基盤は、今後展開される多くのIBMの自動化サービスにおいても安全に利用できるように設計されています。SLAは、特別なスキルを必要とせずWebベースのGUI操作のみで実現可能なソリューションであるため、操作担当者に高いスキルを必要としません。ミスの発生しやすい人手による作業に頼っていたため何段階にも及ぶ承認を経ないと変更作業を実施できない、といった考え方からも徐々に脱却していく必要があります。

5. おわりに

ITインフラ運用を効率化するためには、IBMが取り組んできたコード化のアプローチを利用することが効果的です。自動化対象の優先度を決めてインフラ全体のコードによる管理を進めることで、冪等性のある運用により即応性・柔軟性の高い運用が実現できます。クラウド上の複雑な設定を持つ新しいシステムの自動化をIaCで実現し、また、従来のオンプレミスのシステムではSLAを利用してカタログ・ベースの自動化を実装可能です。

IaCの活用においてはIBMが積極的に採用しているChefを利用することで、サーバーの構築から運用に至るまでのインフラ運用を柔軟にコードで管理することができます。ChefのCookbookの記述は非常に分かりやすく、コードによるインフラの管理を行う入り口としても導入が容易です。SLAはGUIによる操作が可能でありカタログ・ベースの自動化を提供することから、さらに容易にインフラ運用の自動化が実現できます。コンプライアンス対応や頻繁なサーバー構築作業、日々の変更作業と多岐にわたるSLAの自動化ソリューションを活用して理想的なインフラ運用を実現できます。

IaCやSLAの活用によりインフラ運用の自動化を実現

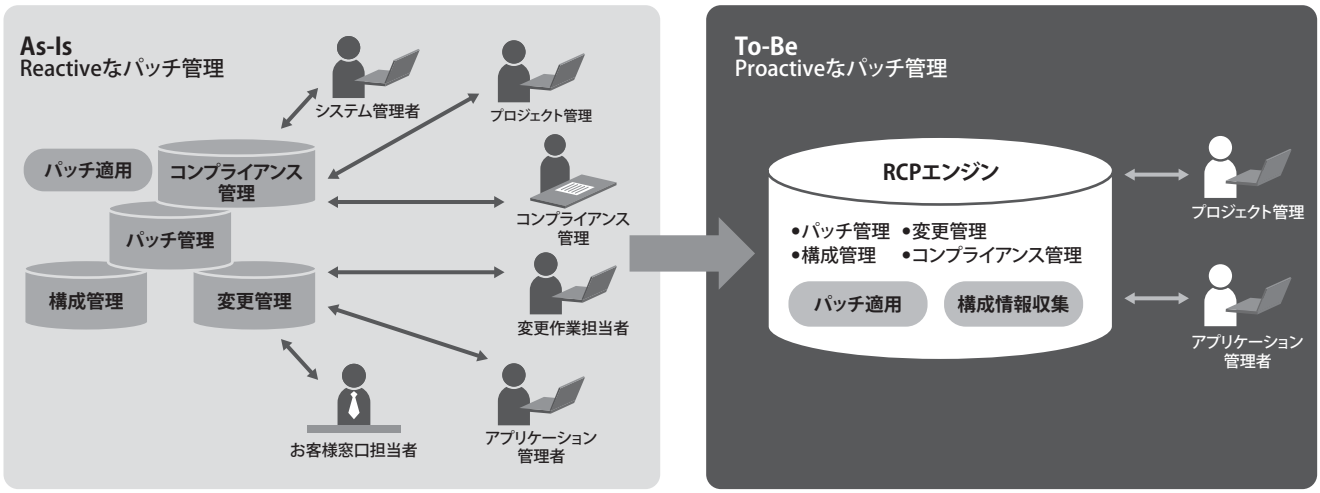

図6. Risk based Continuous Patchによるパッチ管理業務の変化

することで、手作業によるミスの削減や効率化が実現できるだけでなく、より柔軟にシステム拡張や障害対応ができる運用を実現できます。インフラ運用を最大限自動化することで今の時代にふさわしいシステム運用が実現できるのです。

一方で、IaCやSLAによるインフラ運用の自動化にも次のような課題が残されています。

- インフラを管理するためのコードを書ける技術者の育成
- 自動化ではカバーしきれない人手による作業

IBMでは、研修やセミナーを通じた技術者の育成を進めて、コードで管理されたインフラ運用の実現を目指しています。また、IBM Services Platform with Watsonでは、SLAやその他のツール、関連したシステムから膨大なデータがデータレイクに蓄積されます。このデータレイクを活用し分析することで、インフラ運用の自動化が見込まれる領域を正確に判断して、より適切な領域に自動化を適用した世界を実現できます。人間が行う作業は報告書にあがるだけで知識の蓄積にはつながりません。今後自動化したインフラ運用が浸透していくことにより、引き起こされるすべてのイベントやエラー、それに対応した手順などが蓄積され、改善のサイクルが始まっていくことが期待できます。そのための第一歩として、サーバー構築・運用を自動化するためにIaCやSLAを活用してみてはいかがでしょうか。

[参考文献]

[1] 一般社団法人日本情報システム・ユーザー協会：企業IT動向調査2017, http://www.juas.or.jp/cms/media/2017/04/it17_ppt.pdf

[2] Theo Schlossnagle, Luke Kanies, Adam Jacob, et al : Infrastructure as Code, https://www.infoq.com/presentations/infrastructure-as-code

[3] 榊原彰 : 開発と運用の融合 - DevOpsの波 -, https://public.dhe.ibm.com/common/ssi/ecm/co/ja/co112626jpja/industry-corporate-co-magazine-co112626jpja-20180205.pdf

[4] Kief Morris（著）,長尾高弘（訳）,宮下剛輔（監訳）:Infrastructure as Code ―クラウドにおけるサーバー管理の原則とプラクティス,オライリージャパン（2017）

日本アイ・ビー・エム株式会社
グローバル・テクノロジー・サービス事業
アドバンスド・オートメーション
シニア・アーキテクト

青山 真巳
Manami Aoyama

1999年日本IBM入社。流通・製造業のアウトソーシングのお客様システムの設計・構築・運用を経て、クラウド・サービスの企画・開発・設計に従事。2016年よりGTS Automationツールの日本におけるモデリングおよび展開を担当。

日本アイ・ビー・エム株式会社
グローバル・テクノロジー・サービス事業
アドバンスド・オートメーション
アソシエイト・アーキテクト

小山 賢太郎
Kentaro Koyama

2002年日本IBM入社。e-businessホスティング・サービスや金融、メディア分野のお客様システムの設計・構築・運用プロジェクトを経験。2016年より日本IBMグローバル・テクノロジー・サービス事業におけるIaC普及活動に取り組んでいる。

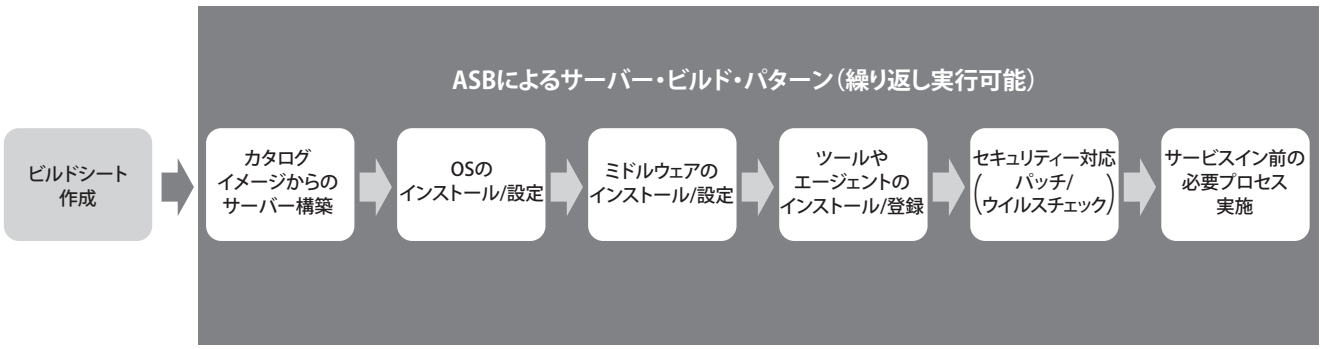

図7. Automated Server Buildによるサーバー構築パターンの例