RED HAT® JBOSS®
MIDDLEWARE

redhat.

WHITEPAPER

JAVA アプリケーションをモダナイズするためのプラットフォーム

クラウドと最新のエコシステムに合わせたワークロードを設計

エグゼクティブサマリー

さまざまな組織が戦略上の岐路に立っています。IDC のアナリストによると、[1]CEO の 3 分の 2 は、デジタルトランスフォーメーションの取り組みを企業戦略の中心に据えようとしています。デジタルトランスフォーメーションの目的は、既存の機能の効率を高めることだけではありません。さらなる大きな目的は、新しいことを新しい方法で行ったり、既存のデータをさらに有効に活用したりすることです。

ところが、現在の IT 支出の 72%[2]は既存のシステムのメンテナンスにあてられており、それが IT スタッフの主な仕事になっています。そのため、今すべきことと、将来のためにすべきこととの間で、大きなジレンマが生じています。

IDC によると、[3] デジタルトランスフォーメーションの戦略の中心になっているのは、ミドル層のアプリケーションです。データ統合、メッセージング、アプリケーションプログラミングインタフェース（API）管理を実施するのは、ミドル層のアプリケーションだからです。ミドル層のアプリケーションは、アプリケーションの開発や管理のプラットフォームにもなっています。従来型のエンタープライズアプリケーションとクラウドネイティブの分散アプリケーションの両方に対応します。

特に、Java ™ EE ベースのアプリケーションプラットフォームは、現在のテクノロジーとクラウドネイティブのアプリケーションの両方をサポートできます。ただし、そのためには以下のことを行う必要があります。

- 既存のスタッフや専門分野の知識を活用した、新しいテクノロジーへの対応
- レガシーアプリケーションと重要なデータの保持
- 既存の環境でも使用できる新しいアプリケーションの並行開発
- 新しいプロセスとアーキテクチャの実装

デジタルトランスフォーメーションの持つ意味は、組織によって異なります。それぞれの組織が独自の戦略や目標を立てるので、そのような意思決定をしたり既存の IT リソースを最も効率良く活用したりするのに役立つな考え方があります。

72% の IT 支出がメンテナンスプロジェクトにあてられています[2]

67% の CEO が、デジタルトランスフォーメーションをビジネス戦略の中心に据えようとしています[1]

59% の CIO が、現在の IT スキルに懸念を抱いています[20]

40% の組織が、今後 3 年以内に IT インフラストラクチャをモダナイズする計画を立てています[18]

1 *Gens, Frank. IDC FutureScape: Worldwide IT Industry 2016 Predictions – Leading Digital Transformation to Scale. IDC, Nov. 2015.*

2 *Zetlin, Minda. "How to Balance Maintenance and IT Innovation." ComputerWorld, 21 Oct. 2013. Web.*

3 *Fleming, Maureen. New Middle-Tier Competencies Enabling Digital Transformation. Rep. IDC, June 2016. Web.*
提供元：Red Hat

facebook.com/redhatjapan
@redhatjapan
linkedin.com/company/red-hat

jp.redhat.com

未来を見据える: プラットフォーム、プロセス、アーキテクチャ

IDC のアナリストである Al Hilwa 氏は、ソフトウェア開発の現状に関するプレゼンテーションの中で、[4] 優れたソフトウェアは、ソフトウェアのアーキテクチャ、開発者のプロセス、開発者のスキルの結晶であると述べました。IT チームと運用チームは、その 3 つすべての分野で大きな変化に直面しています。クラウド型の分散コンピューティングによって、インフラストラクチャの規模に関する新しいエコノミーが導入され、クラウド環境のメリットを活用できるように 3 つの基本要素（アーキテクチャ、プロセス、プラットフォーム）が急速に進化しています。

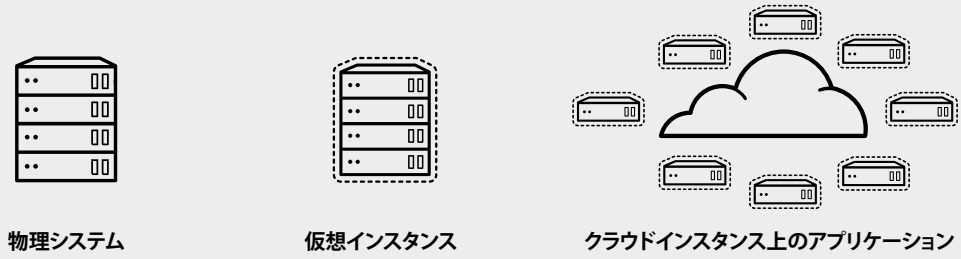

図 1. 物理、仮想、クラウドのトポロジー

プラットフォーム: クラウド

クラウドコンピューティングは、次世代の IT イノベーションにおける中核的なインフラストラクチャになると予想されています。アナリストグループである IDC は、2020 年までに、インフラストラクチャに関する総支出の最大 70% がクラウドサービスにあてられるようになると予測しています。[5]

クラウドが効率的である大きな理由の 1 つは、環境の変化に対応しやすいということです。仮想化が IT の大きなテーマになっているのは、運用環境を物理的な環境から取り出して抽象化できるからです。仮想化によって、オペレーティングシステムを物理システムから完全に分離し、複数のオペレーティングシステムインスタンスを同じハードウェアにインストールして実行できるようになります。クラウドを導入すれば、環境をさらに細分化し、実行するアプリケーションを基盤となるオペレーティングシステムや物理的な環境から取り出して抽象化できます。

クラウドベースのサービスは、1 つの物理システムに厳密に割り振るのではなく、数多くの分散ノードに配置できます。これにより、1 つのノードで障害が発生しても別のノードで運用を続けることが可能になり（冗長性の確保）、サービスの需要に基づいてインスタンスを新しく作成、破棄することも可能になります（柔軟なスケーラビリティ）。

このような軽量のインフラストラクチャによって、高分散型のアーキテクチャパターン（マイクロサービスなど）に対応できるようになります。クラウド環境に移行すれば、リソースを最大限に活用して運用効率の向上やコストの削減を実現することも可能です。

4 Hilwa, Al. "The New Developer Landscape – Understanding the Modern Software Developer." Mar. 2016. IDC event presentation

5 Ibid., Gens.

プロセス: DevOps とアジャイル

アジャイル、継続的インテグレーション/継続的デリバリー（CI/CD）、そして DevOps は、モダナイズした IT チームの構成や機能に欠かせない中心的なコンセプトであり、それぞれが相互に関連し合っています。

アジャイルは、プロジェクトの計画と実行の手法です。アジャイルソフトウェア開発宣言[6]によると、アジャイルの手法には 4 つの基本原則があります。

- チーム第一
- 外部グループとの協調
- 状況の変化への対応
- 分かりやすいソフトウェアの作成

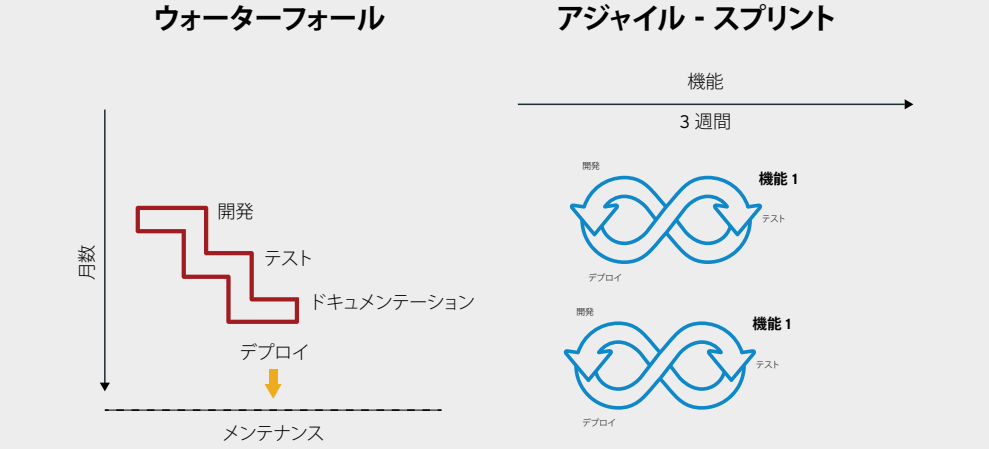

図 2. ウォーターフォールとアジャイルの開発サイクル

アジャイルプロセスでは、プロジェクトの業務を管理可能なタスクに分割します。目標は、タスクを短時間で繰り返し、実際に機能するソフトウェアを機能単位で提供していくことです。この考え方は、厳密に順序を定めるウォーターフォールプロジェクトの考え方と対極にあります。最近のガートナーの分析によると、[7]アジャイル手法がかなり広まった結果、ウォーターフォールプロジェクトは 50% を下回っています。

アジャイル手法の導入でプロジェクト開発に生じた変化の 1 つは、他のグループとの協調です。それも、開発完了後ではなく、計画と開発のプロセス全体での協調です。スプリントの最終段階で、ソフトウェアは十分に機能する状態になっていなければなりません。つまり、開発とテストを並行して行う必要があります。アジャイルチームは、開発サイクルの中にテストを組み込みます。このような継続的なテストの考え方が継続的インテグレーション（CI）に発展し、短時間でのリリースという考え方が継続的デプロイ（CD）に発展しました。これは、各段階が明確に分離したプロセスではなく、継続的なプロセスです。

6 Beck, Kent, et al. 2001. "Manifesto for Agile Software Development". Agile Alliance.

7 Wilson, Nathan. Modernizing Application Development Primer for 2016. Gartner, 14 Jan. 2016. Web.

ただし、アプリケーションライフサイクルはデリバリーの時点で終わるわけではありません。アプリケーションには、デプロイやメンテナンスが必要です。その種の作業は基本的に、開発チームではなく運用チームや IT チームが担当します。チーム間の協力関係が確立していなければ、開発チームは運用環境が分からず、運用チームは製品の戦略目的を理解していない、といった状況が生じかねません。理解にギャップがある環境では、ソフトウェアやインフラストラクチャが本来の目的どおりに機能しなくなる事態になりかねません。そこで、開発チームと運用チームの協調を目指す考え方が生まれました。それが DevOps です。

アーキテクチャ: マイクロサービス

従来のエンタープライズソフトウェアは大規模なものでした。1 つのアプリケーションで、エンタープライズ全体のピーク時の負荷に対応しなければなりませんでした。そのアプリケーションで必要な機能はすべて、そのアプリケーションが実行しなければなりません。各種サービスは、モノリシックなアプリケーションの機能にすぎませんでした。データベースは、モノリシックなアプリケーションの良い例です。1 つのデータベースで組織全体のニーズに対応します。

エンタープライズコンピューティングのニーズが複雑になるにつれ、モノリシックなアプリケーションのメンテナンスの必要性が高まりました。モノリシックなインフラストラクチャの場合、小さな変更も困難な場合があります。一部の変更であっても、全体を変更しなければならないからです。そこで登場したのが、サービス指向アーキテクチャ（SOA）という新しい手法です。サービス指向アーキテクチャでは、1 つのアプリケーションですべてに対応する代わりに、複数のアプリケーションによって複数の機能を提供します。そして、それらのアプリケーションをエンタープライズサービスバス（ESB）のような統合パターンによって疎結合します。

とはいえ、SOA を導入すると、システム環境全体が複雑になります。アーキテクチャの新しいコンポーネントの導入やアップグレードの実施は容易になりますが、コンポーネントの相互作用を明確に理解していないと、各種の変更が環境全体に連鎖的に波及してしまうリスクがあります。

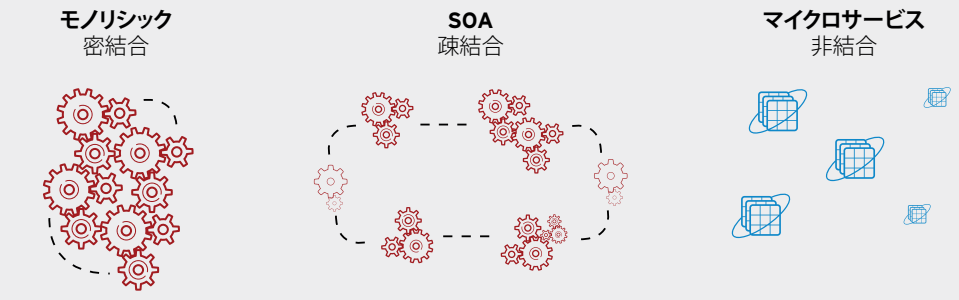

図 3. 各種アーキテクチャにおけるサービスの関係

SOA の方向性は正しかったとはいえ、それをさらに発展させたマイクロサービスというアーキテクチャが登場しました。マクロサービスアーキテクチャは、特殊化したサービスを疎結合するという点で SOA のパターンとよく似ていますが、細分化をさらに推し進めています。マイクロサービスアーキテクチャでは、次のようにサービスを大変明確に定義しています。

ナンバーワン

「Java の未来は明るいでしょう。言語もプラットフォームも、一晩で変わるわけではありません。優れたテクノロジーはオープンソースですが、今すぐ Java を脅かすようなものはありません。マイクロサービスの現状を見ても、その多くは Java ベースのものです。エンタープライズ Java は再定義され、オープンハイブリッドクラウドコンピューティングによる、この新しいパラダイムで作り変えられています。素晴らしい実績のすべてはオープンソースで行われているのです」

RICH SHARPLES、
ミドルウェア製品開発
シニアディレクター、
RED HAT

- 1 つの明確な目的
- 十分に定義されたパラメータ
- 多言語での実装

このアーキテクチャのサービスでは、Representational State Transfer（REST）API のような共通のメッセージングフレームワークを使用して相互に対話します。難しいデータ変換トランザクションや統合のための層は必要ありません。

このメッセージングフレームワークを使用すれば、新しい機能やアップデートを短時間で提供できるようになります。それぞれのサービスは独立しています。1 つのサービスの置き換え、拡張、削除がアーキテクチャ内の他のサービスに影響することはありません。このような軽量のアーキテクチャによって、分散型またはクラウドのリソースを最適化し、個々のサービスの動的なスケーラビリティを確保できます。

Conway の法則と未来

クラウド、DevOps、マイクロサービスには、共通する 1 つの特性があります。それは、分散環境の複雑さです。1967 年に、ソフトウェア開発者の Melvin Conway[8] は、ソフトウェアの設計パターンは、開発に関わるチーム間のコミュニケーションの構造とよく似ていると述べました。コミュニケーションの柔軟性が乏しかったり、不透明または不十分であったりすると、実行されるソフトウェアも貧弱になる、というわけです。

組織がデジタルトランスフォーメーションに向けたどの段階にあるとしても、その最終結果には必ず IT チームのカルチャーが反映されます。したがって、テクノロジーの問題とは別に、チームのカルチャー[9]について、以下の点に取り組む必要があります。

- チームを越えてだれもが深く理解できるコミュニケーションパターンを確立する
- チーム間の壁を最小化する
- インフラストラクチャと人的交流を柔軟にする

複雑な分散型のアーキテクチャであるマイクロサービスには、しっかりとした基礎が必要です。

テクノロジーの面から言えば、マイクロサービスにはクラウドベースのインフラストラクチャが不可欠です。[10] 一方、組織の面から言えば、DevOps やアジャイルのしっかりした作業環境がなければ、つまり、役割を越えたチーム間の協力関係やコミュニケーションが確立していなければ、マイクロサービスは成り立ちません。

JAVA EE の過去と未来

今後クラウドベースの軽量なアーキテクチャによって、生産性だけでなく複雑性も増すとなると、そうした状況に現在の IT チームをどのように備えさせることができるしょうか？

1995 年に登場した Java は、現在では世界で最も普及したプログラミング言語になっています。[11] Oracle は、世界中の Java 開発者の数を 900 万人以上と見積もっています。[12] これは、世界で 1,100 万人にのぼるプロの開発者の 82% ほどに相当します。[13]

---

8 Conway, Melvin E. "How do Committees Invent?" Datamation, 14 (5): 28-31. April 1968.

9 DevNation Afternoon General Session. By Rachel Laycock. California, San Francisco. 27 June 2016. Performance. https://youtu.be/EC2rk9Jh5Ps

10 Gens, Frank. IDC FutureScape: Worldwide IT Industry 2016 Predictions – Leading Digital Transformation to Scale. IDC, Nov. 2015.

11 "TIOBE Index for August 2016." Aug. 2016 Web.

12 Beneke, Timothy and Tori Wieldt. JavaOne 2013 Review: Java Takes on the Internet of Things.

13 Hilwa, Al. "The New Developer Landscape – Understanding the Modern Software Developer." Mar. 2016. IDC event presentation.

Sun（後の Oracle）は、Java プログラミング言語を基礎にして、一般的な操作の標準、API、アプリケーションのランタイム環境を構築しました。それを総合したものが Java Enterprise Edition（Java EE）です。Java EE の仕様を実装したサーバーが Java アプリケーションプラットフォームであり、そのアプリケーションプラットフォームが数多くの IT 開発環境の中核になっています。Java のサーバー/クライアント型のプログラミングは、初期のインターネットフレームワークにも、その後のエンタープライズレベルのアプリケーションにも実に適していました。

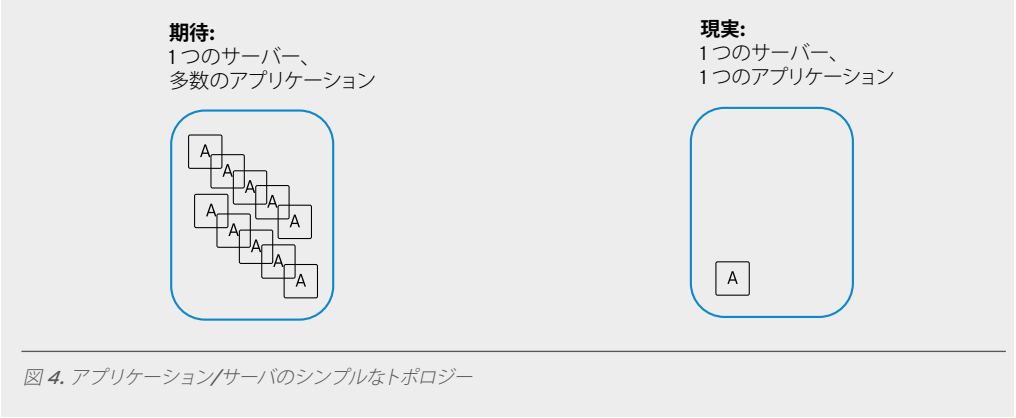

図 4. アプリケーション/サーバのシンプルなトポロジー

アプリケーションプラットフォームが開発されていた 2000 年代の初期には、ほとんどの IT アーキテクチャがモノリシックなアプリケーション上に成り立っていました。Java EE のアプリケーションプラットフォームは、1 つの中心的な場所で多数の Java アプリケーションをホストするよう見込まれていました。しかし実際には、トラフィックの分散、ネットワークの帯域幅やレイテンシー、冗長性、組織の縦割り構造などが原因で、1 つのアプリケーションプラットフォームで 1 つのアプリケーションだけをホストする、といった状況が多く見られました。ガートナーは、[14] そのような Java アプリケーションプラットフォームのことを「スーパープラットフォーム」と呼んでいます。この状況は、多くの場合 IT リソースの浪費につながりました。

Java が言語としてもプラットフォームとしても強靭な理由の 1 つは、適応性の高さです。Java と Java EE は、Java Community Process というコミュニティで定義されています。この Java コミュニティは、Java EE の開発にも、関連するコミュニティプロジェクト（MicroProfile、Wildfly Swarm、Node.js など）にも積極的に関わっています。

このコミュニティでは、Java および Java プラットフォームにおいて、ネイティブなクラウドの開発や軽量で相互に連結されたアプリの展開が可能となる方向を目指して活動しています。Java EE 6 では、プロファイルというコンセプトが導入され、従来の大規模な Web サーバーに対応するフルプロファイルと、軽量のアプリケーションに対応した Web プロファイルの両方が用意されました。Java EE 7 では、プロファイルとモジュール化のコンセプトがさらに拡張され、いくつかのコミュニティプロジェクトがそれを推し進めています。

- MicroProfile では、Web プロファイルより小さく、マイクロサービスにより適した新しい Java プロファイルの仕様を検討しています。トランザクションやメッセージングなどの機能に特化したプロファイルです。
- Wildfly Swarm では、Java アプリケーションを 1 つのコンテナイメージとして取り扱い、必要最小限のライブラリや前提コンポーネントを 1 つの fat jar ファイル（標準的な Java アーカイブ）としてバンドルしています。

14 Wilson, Nathan. Modernizing Application Development Primer for 2016. Gartner, 14 Jan. 2016. Web.

モダナイゼーションに向けて

- クラウドへの移行
- クラウドやデータセンターで同等のパフォーマンスが得られるアプリケーションプラットフォームの使用
- ビジネス戦略に基づいた IT 戦略の推進
- インフラストラクチャでモダナイズする部分およびその方法を決定
- チームとカルチャーの構築
- 開発者へ優れたツールを提供
- 統合、スケーラビリティ、相互運用に対応した IT エコシステムの設計

Java も他の多くのテクノロジーと同じようにクラウドインフラストラクチャに適応していますが、Java には 1 つの大きな強みがあります。その中核的な言語のテクノロジーに、数百万人もの開発者がすでに精通しているということです。アプリケーションのアーキテクチャや実装は新しくても、それを作成するために必要なスキルはすでに確立されています。

リソースを最大限に活用するための変化

IDC は、クラウドのことを第 3 のプラットフォーム IT 環境の中核と呼んでいます。[15] IDC では、コンピューティングの進化を 3 つの段階に分けています。[16]

1. メインフレームとパーソナルコンピュータ
2. インターネットベースのトランザクションとクライアント/サーバー型のアーキテクチャ
3. クラウドでホストするアプリケーション中心のテクノロジー（モバイル、ソーシャル、IoT、ビッグデータなど）

図 5. テクノロジープラットフォームの段階

この第 3 のプラットフォームは他の 2 つに大きく依存していますが、単に同じことをより効率的に行うのではありません。その 2 つの層を基盤として新しい事を行うのです。

1. クラウドへの移行（Java EE）

モダナイズした IT 環境の中核にあるのは、クラウドコンピューティングです。その理由は、クラウドコンピューティングのスケーラビリティにあります。特に、従来型の物理システムや仮想システムでは不可能だった方法で、その時々の需要に応じて動的にノードを追加したり削除することができますy。

Java EE などのテクノロジープラットフォームでは、次のようにさまざまな環境に対応することが不可欠です。

15 Gens, Frank. IDC FutureScape: Worldwide IT Industry 2016 Predictions – Leading Digital Transformation to Scale. IDC, Nov. 2015.

16 IDC. IDC Predicts the Emergence of "the DX Economy" in a Critical Period of Widespread Digital Transformation and Massive Scale Up of 3rd Platform Technologies in Every Industry. 4 Nov. 2015. Web.

- オンプレミス
- パブリッククラウド（Amazon Web Services、Microsoft Azure、Google など）
- プライベートクラウド（OpenStack® プライベートクラウドなど）
- コンテナ
- ホステッドサービス

アプリケーションは、どの環境でも同じように稼働しなければなりません。アプリケーションプラットフォームによっては、機能しない環境があるものや、あらゆる環境で同じ機能を提供できないものもあります。適切な Java EE プラットフォームを選択すれば、混合環境で不可欠な相互運用を実現できます。

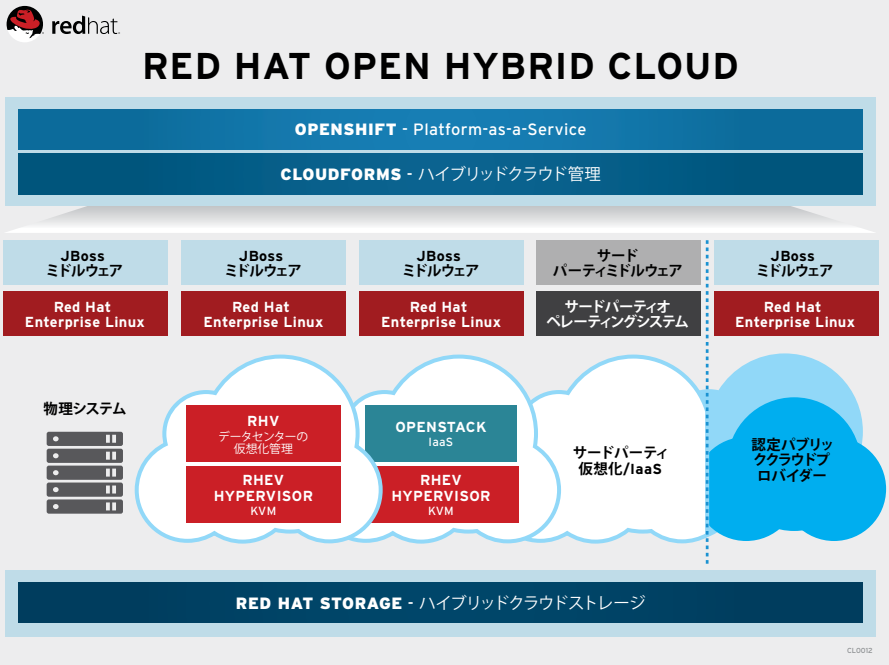

図 6. ハイブリッドクラウド環境の概要

ハイブリッド環境では、運用が複雑になるだけでなく、さまざまな環境で運用するためのコストもかかります。ガートナーは、IT 部門のコスト削減可能な分野の 1 つとしてソフトウェアライセンスを挙げており、[17] コストの削減において最も見過ごされがちな分野であると述べています。ソフトウェアの価格設定は複雑であり、オンプレミス、仮想、クラウドで別々のライセンスが必要な場合や、サポートの種類ごとに別々のサブスクリプションサービスが設定されている場合があります。

17 McGittigan, Jim, and Sanil Solanki. The Gartner Top 10 Recommended IT Cost Optimization Ideas, 2016. Tech. no. G00301094. Gartner, 29 Feb. 2016. Web.

ヒント：移行の規模は組織に合わせて決定

デジタルトランスフォーメーションの目標を達成するために、すべての IT インフラストラクチャをスタートアップスタイルのマイクロサービスアーキテクチャに移行しなければならないわけではありません。戦略的なプランをどう立てるかによっては、クラウドへの移行、コンテナベースのデプロイ、または新しい IoT やモバイルイニシアチブの導入でも戦略的なプランとして成立するかもしれません。

2. 既存のリソースの確認

それぞれの環境に合った最適なオプションを見定めるには、変化のリスクと戦略的な目標とのバランスを考える必要があります。要件は組織によって異なります。デジタル戦略はビジネス戦略を反映していなければなりません。

- 組織の目標や戦略的な方向性はどのようなものでしょうか？
- その方向に進むために必要なスキルやリソースがありますか？
- そのようなアプリケーションに対応できる環境がありますか？

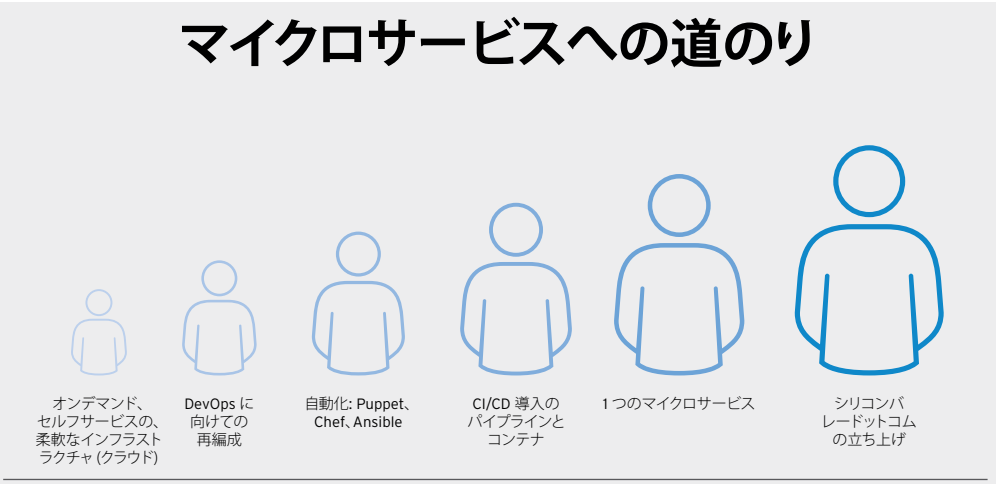

図 7. 環境の進化における段階

3. モダナイゼーションを行う対象と方法の決定

IDC の Peter Marston 氏によると、デジタルトランスフォーメーションの中心的な問題は、アプリケーションのモダナイゼーションへの取り組み方です。その調査結果によると、約 40% の組織が、アプリケーションのモダナイゼーションを IT の最優先事項にしているか、今後 3 年以内にそうすると回答しています。[18]

モダナイゼーションに取り組むいくつかの方法を以下にまとめます。

- 改良/再設計した既存のアプリケーションを、モダナイズした環境に取り込む
- 1 つの環境から別の環境にアプリケーションを移行する
- 既存のアプリケーションを新しいアプリケーションに置き換える
- 並列的な環境を作成する

どの方法を選択するかは、組織でマイクロサービスへの移行をどこまで進めるかによって異なります。

1 つの目標は、IT 部門がメンテナンスにかける手間やコストを減らすことです。IT スタッフの手間やコストを、新しいプロジェクトと既存のプロジェクトの間で均等に配分できるなら理想的です。とはいえ、IT 支出の 72% がメンテナンスと運用のプロジェクトにあてられているのが現状です。CTO の大半 (63%) が、その比重を低くしたいと考えています。[19]

18 Marston, Peter. Ten Criteria to Use for Application Modernization Service Provider Selection. Rep. no. IDC #US41012716. IDC, Feb. 2016. Web.

19 Zetlin, Minda. "How to Balance Maintenance and IT Innovation." ComputerWorld, 21 Oct. 2013. Web.

クラウドベースの Java EE の基盤は、メンテナンスコストの削減に役立ちます。既存のアプリケーションと新しいアプリケーションの両方を同じ基盤の上で実行できるからです。一方がモノリシックで、他方がマイクロサービスであっても、それが可能です。さらに重要な点として、Java アプリケーションは、最終的に別の環境に移すことができます。

この可搬性により、アプリケーションをモダナイズする経路を確保しながら、トランザクションのコストや移行に伴うリスクを軽減できます。その結果、組織でインフラストラクチャを段階的に変更していくことが可能になります。まずクラウドの導入によってコストを削減してから、次にコンテナの導入によって運用効率を高める、といった具合です。組織の戦略に合わなければ、マイクロサービスや高分散型のアーキテクチャにこだわる必要はありません。

4. チームの改革

CIO の直面する大きな課題の 1 つは、新しい第 3 のプラットフォームに対応できる IT スタッフの確保です。IT 部門の人材難に関する Harvard Business Review の分析レポートによると、59% の CIO が、IT や戦略的な課題への対応を阻む原因に、スキル不足があると感じています。[20] IDC の言う第 3 のプラットフォームに関して懸念されていのは、以下の分野です。

- ビッグデータと分析（36%）
- アーキテクト: エンタープライズ（27%）、テクニカル（24%）
- 開発（27%）
- モバイル開発（24%）
- IT 戦略（22%）

Forrester Research の Nigel Fenwick 氏は、IT 組織はカスタムソリューションによって IT の目的を達成しようとすることが多いものの、それは必ずしも最善の戦略的な方法ではないと指摘し、次のように述べています。「汎用的な機能をサポートするためのソフトウェアのカスタマイズに、巨額の費用が投じられています。その結果、IT が複雑になり、インターフェースの操作が難しくなり、IT の俊敏性が損なわれ、コストがかさんでいます」。[21]

あらゆる機能を実装しようとする代わりに、簡略化して IT 戦略に集中することができます。そのために、次のような方法があります。

- 組織の戦略的な目標に合わせて、必要なコア機能を 2 つか 3 つに絞り込む
- 他のすべての要件については、カスタムソリューションではなく標準ベースのソリューションを使用する
- メンテナンスの容易なソリューションを選択する

CIO がスキルの不足を最も感じているのは、テクノロジー関連のスキルではなくプロセス関連のスキルです。オープンスタンダードを使用すれば、テクノロジーの習得にかかる時間を短縮できます。Java のような一般的なテクノロジーであれば、なおさらです。開発用のプラットフォームとして Java EE のアプリケーションプラットフォームを実装すれば、IT 部門は、既存の知識や経験を活かして新しい分野のプロジェクトに取り組むことができます。これにより、人材発掘の余地も広がります。Java は、プロの開発者のコミュニティではごく一般的なスキルだからです。

5. 開発者への優れたツールの提供

テクノロジープラットフォームに開発者向けの統合ツールを用意することで、開発サイクル全体をスムーズに進めることができます。特に、テストや自動化のシステムを組み込んでおくのが得策です。開発者が重宝するツールの主な分野には、次のようなものがあります。

---

20 I.T. Talent Crisis: Proven Advice from CIOs and HR Leaders. Tech. Harvard Business Review Analytic Services, July 2016. Web. Sponsored by Red Hat

21 Zetlin, Minda. "How to Balance Maintenance and IT Innovation." ComputerWorld, 21 Oct. 2013. Web.

- CI/CD に対応した統合テストモジュール
- デプロイの自動化
- 開発者用ツールキット
- カスタムクラスのロード
- パフォーマンス

ごく小さなパフォーマンス機能でも、大きな影響を与えます。例えば、アプリケーションのデプロイ時の始動時間を短くできれば、週に数時間の節約につながり、開発者の生産性を向上できるかもしれません。開発者は、通常の開発作業で 1 日に何度もアプリケーションを再始動することがあるからです。

DevOps や CI/CD では、テスト機能を組み込むことも不可欠です。オーストラリアのある IT 企業は、デプロイプロセスにテストモジュールを組み込むだけで、開発者の生産性を 15% アップさせることができました。[22] 開発、テスト、運用を統合することで、コードの質を上げたり開発サイクルを短縮することができます。

6. エコシステムの構築

IDC のアナリストである Maureen Fleming 氏は、デジタルトランスフォーメーションの強みは統合にあると述べています。[23] 第 3 のプラットフォームでは、以下のような機能を統合できます。

- 仮想化、パブリッククラウド、プライベートクラウド
- コンテナとオーケストレーション
- データの仮想化
- メモリーキャッシュとストレージ
- 複数のメッセージングプロトコル
- さまざまなデータソースの異なるデータ形式
- 管理とデプロイのツール
- テストの自動化
- ビジネスプロセスの自動化

デジタルトランスフォーメーションのプラットフォームを設計する上で、実際のプラットフォームはその取り組みの一部にすぎません。プラットフォームやそのプラットフォームで実行するアプリケーションを統合した、大きなエコシステムを構築する必要があります。

IDC の Peter Martson 氏は、アプリケーションのモダナイゼーションに関するより大きな戦略の一部として、アプリケーションプラットフォームプロバイダーの広範なエコシステムを検討するよう勧めています。[24] テクノロジーやスキルに関する経験や指針を提供できるソリューションプロバイダーは、アーキテクチャの計画からクラウドのプロビジョニングにいたるまで、さまざまな分野でサポートすることができます。

---

22 Fleming, Maureen, and Matthew Marden. The Business Value of Red Hat JBoss Enterprise Application Platform. Tech. no. #257256. IDC, July 2015. Web. Sponsored by Red Hat

23 Fleming, Maureen. Integration Is a Core Competency of Digital Transformation. Tech. no. IDC #US41293916. IDC, May 2016. Web. Sponsored by Red Hat

24 Marston, Peter. Ten Criteria to Use for Application Modernization Service Provider Selection. Rep. no. IDC #US41012716. IDC, Feb. 2016. Web.

まとめ

デジタルトランスフォーメーションは、IT のインフラストラクチャやデータからお客様向けの製品を作り出していくための戦略的な手法です。デジタルトランスフォーメーションを推進するには、内部用のアプリケーションの開発、または内部のビジネス要件への対応といった発想から抜け出して、そのアプリケーションを製品化することや、データやデータソースを利用する新しい方法を考え出すこと、お客様に働きかける新しい方法を編み出すことを目指す必要があります。

デジタルトランスフォーメーションを進めるには、以下の 3 つが必要です。

- 組織の明確な目標と戦略
- チームを越えた、協調やコミュニケーションのためのしっかりしたプロセス
- だれもがよく理解できる統制の取れた分散アーキテクチャ

Java アプリケーションは、この 20 年間、エンタープライズの中心的なテクノロジーとしての地位を固めてきました。企業にとって、その蓄積されたデータや機能や知識は極めて重要です。Java ベースのアプリケーションを利用すれば、最先端の開発手法で既存のアプリケーションにも対応しながら、クラウドネイティブのアーキテクチャに対応したプラットフォームを構築することが可能になります。そのようにして、従来型のモノリシックなエンタープライズアプリケーション開発とクラウドベースのアプリケーション開発を組み合わせれば、既存の知識やリソースを活用しながら、新しいアプリケーションモデルへの移行を積極的に推進することができます。

クラウドとの親和性の高い Java アプリケーションプラットフォームには、次のような利点があります。

- IT 部門のスキル不足やテクノロジー不足の影響を緩和できる
- 人材発掘の幅が広がる
- 既存のワークロードの移行戦略を展開しながら、新しい環境での開発に取り組むことができる
- 統合やデータ管理などの戦略的な取り組みに役立つ他のテクノロジーを導入するための環境を準備できる

最適な Java アプリケーションプラットフォームによって可能になるのは、既存のエンタープライズアプリケーションの管理だけではありません。既存の IT リソースを最大限に活用し、必要なメンテナンスプロジェクトを継続しながら、経営陣が計画しているデジタルトランスフォーメーションを実現するための手段になるのです。

RED HAT について

オープンソースソリューションのプロバイダーとして世界をリードする Red Hat は、コミュニティとの協業により高い信頼性と性能を備えるクラウド、Linux、ミドルウェア、ストレージおよび仮想化テクノロジーを提供、さらにサポート、トレーニング、コンサルティングサービスも提供しています。Red Hat は、お客様、パートナーおよびオープンソースコミュニティのグローバルネットワークの中核として、成長のためにリソースを解放し、ITの将来に向けた 革新的なテクノロジーの創出を支援しています。

アジア太平洋
+65 6490 4200

インドネシア
001 803 440224

ニュージーランド
0800 450 503

ベトナム
800 862 6691

オーストラリア
1 800 733 428

日本
03 5798 8510

フィリピン
800 1441 0229

中国
800 810 2100

ブルネイ / カンボジア
800 862 6691

韓国
080 708 0880

シンガポール
800 448 1430

香港
852 3002 1362

インド
+91 22 3987 8888

マレーシア
1 800 812 678

タイ
001 800 441 6039

台湾
0800 666 052

facebook.com/redhatjapan
@redhatjapan
linkedin.com/company/red-hat

jp.redhat.com
INC0441724_1016