解 説 2

サービス事業を支えるクラウド技術

― 人が集まるオンライン・ビジネスの裏側を探る ―

近年サービス事業はあらゆる産業の競争力の源泉として重要度を増しており、とりわけコスト、スピード、多様性、オンデマンド性の観点から IT を駆使したサービスへの期待が高まっています。本稿では高いユーザー・エクスペリエンスを実現するサービスの機能要件と非機能要件を概観し、継続的改善とライフサイクルに合ったインフラ選択の重要性を指摘します。続いて付加価値の高いサービスを開発・運用するためのクラウド・コンピューティング（以下、クラウド）の利用技術について、IaaS（Infrastructure as a Service）、PaaS（Platform as a Service）のそれぞれについてプライベート・クラウド、パブリック・クラウドの両方の視点から解説します。継続的改善が高い頻度で（毎日、毎週）行われる Web 上のサービスは、開発環境から運用環境へのデプロイが頻繁に発生し、またサービス・ライフサイクルにおいてはインフラ要件の変化にタイムリーに対応する必要があります。どちらもサービスのクラウド環境間の可搬性（ポータビリティー）の問題に帰結しますが、このようなインフラにロックインされない PaaS は IaaS 上に各コンポーネントを疎結合で実装することにより実現が可能になります。本稿ではこのような問題への対応方法を PaaS、IaaS での実装例を用いて解説いたします。また IBM が発表した IBM PureApplication System（以下、PureApplication System）が示す次世代の PaaS の在り方についても示します。

❶ はじめに

わが国の産業構造の変化の歴史の中で第三次産業であるサービス業はその経済規模、雇用への影響力において重要度を増し続けていますが、近年の特徴として、第一次産業、第二次産業の付加価値や国際競争力を高めるサービスの効用が一層注目されているという点が挙げられます。特にコスト競争力とグローバル展開の観点から、IT を利用したインターネット上のサービスに期待が集まっています。従来の産業区分にとらわれず、あらゆる業種、業態で行われるため、ここではサービス事業と呼ぶことにします。また本稿では説明を簡単にするために Web アプリケーションによるサービス事業を前提に考えますが、基本的に IT によるサービス全般について普遍的な内容となっています。

サービスの成長の源泉はサービスの受け手の満足度です。例えば運輸、保険、小売、ホテルなどで、サービスの利用者は期待通りのサービスが提供されれば満足し、期待を大きく上回ったときは感動します。期待とコストは連動するので、コストを抑えて期待をマネージした上で、それを大きく上回るサービスを提供するのがサービス事業成功の王道と考えられますが、これは IT によるサービスでも同じです。提供されるサービスの優劣はサービス品質やユーザー・エクスペリエンスで評価されますが、「いいサービスだった、また利用したい」というユーザー視点がより重視される傾向にあります。

IT サービスのユーザー・エクスペリエンスを高めるためには優れたコンテンツを最適なインフラから提供することが必要です。優れたコンテンツは機能要件（Functional Requirement：以下、FR）、最適なインフラは非機能要件（Non-functional Requirement：以下、NFR）を満たす手段となります。サービス事業は変化が激しいので FR、NFR の変化にどれだけダイナミックに対応できるかが成功の鍵となります。

例えば商品の販売を目的とした Web サイトがあるとします。A 社は商品軸で機能説明を行うので更新頻度は商品のライフサイクルに合った緩やかなものです。一方 B 社は利用シーンを前面に出したプロモーションを行っており、最新の流行やソーシャル・ネットワークからのフィードバックを反映して頻繁にコンテンツを更新します。また B 社はコンテンツを入れ替えるだけでなく、新たなリコメンデーション・エンジンの導入など、Web サイトの仕組み自体も継続的に改善し続けます。これはデパートが商品の入れ替えのみならず店舗の大規模な模様替えを定期的に行うのと似ています。A 社、B 社どちらのサイトが利用者にとって魅力的でより多くの人を引き寄せるかは明らかです。B 社のようなサービスを立ち上げ、高いユーザー・エクスペリエンスを維持するには、利用者からのフィードバック（苦情、賞賛、

表1. サービス・ライフサイクルとインフラ要件

要求項目 / フェーズ	ITインフラ・キャピタル投資	ITインフラ運用経費	ITインフラ変化対応のスピード	ITインフラ品質（SLA）
商品開発期	ゼロ～最小	短期最適化	最優先	低
パイロット展開期	ゼロ～最小	短期最適化	高	スピード・コスト優先
急成長期	運用経費との最適化	中期最適化	拡大・縮小・撤退容易性	中～高（適材適所）
安定成長期	運用経費との最適化	長期最適化	季節変動、グローバル展開など	中～高（適材適所）

最適なインフラを適材適所に配置できるハイブリッド・クラウドが理想的。
その場合、プラットフォーム間の互換性やワークロードのポータビリティーが鍵になる。

質問、要望）をサービス内容、すなわちWebアプリケーションに素早く反映することが必要です。そこで求められるのは効率的なプログラム開発と、それを安定してデリバリーするための運用との連携です。製品開発では出荷をもって開発部門の仕事は一段落となりますが、B社のようなサービス事業ではサービスイン後も開発部門の仕事は終わりません。高いユーザー・エクスペリエンスを目指すという意味では、そこから本当の開発が始まるといってもよいくらいです。「永遠のベータ版」という言葉がありますが、これは継続的な改善へのコミットメントを表しています。

NFRはサービスの性能に関係します。今C社、D社が同時にWeb上でサービスを立ち上げたとします。両社とも優れたコンテンツを提供し、それを維持する努力も欠かさなかったためユーザーが増えてきました。しかしその結果C社のサービスはレスポンスが悪くなり、利用のピーク時間帯にはアクセスが困難になってしまいました。一方のD社はユーザー数の増加や利用のピークに合わせて最適なインフラに切り替えていったため、サービス事業の成功がユーザー・エクスペリエンスの低下を招かずに済みました。それどころかD社は運用コストの削減とネットワーク・セキュリティーの強化を実現し、基幹サービス事業の盤石な基盤を実現しました。両社の違いは、サービスのライフサイクルに合わせてインフラを切り替える柔軟性の有無にあります。サービス事業は試行錯誤の連続ともいえます。サービス立ち上げ時には迅速な市場参入とともにうまくいかなかった場合の速やかな撤退についても考慮しておく必要があります。一方で大ヒットは突然やってきますし、成功したサービスはインフラに対してより高いサービス・レベルと長期的視点での低コストを要求します。**表1**はサービス・ライフサイクルと最適なインフラ要件を示していますが、重要なのはサービスの成長に伴いWebアプリケーションやミドルウェアを最適なインフラに短期間で切り替えられることです。別な表現をすればサービスに可搬性（ポータビリティー）があるということです。

ITによるサービス事業の成功要因について見てきましたが、次にクラウド・コンピューティングによって何がどう変わるかを考えていきます。

❷ サービス事業を支えるIaaS (Infrastructure as a Service) 技術 [1]

IaaSでは、仮想マシン（Virtual Machine：以下、VM）の元になる定義体はイメージ（仮想アプライアンス）、実行環境で稼働するVMはインスタンスと呼ばれます。イメージがあればセルフサービス・ポータルからインスタンスを短時間でプロビジョニングすることができます。サービス事業におけるコンテンツ（本稿ではWebアプリケーション）を開発するとはこのイメージを開発することにほかなりません。

イメージの内容はOSやソフトウェアのバイナリー・データとプロビジョニング時に用いられるイメージの内容はOSやソフトウェアのバイナリー・データとプロビジョニング時に用いられる管理情報がから成ります［2］。IBMのクラウド・コンピューティング製品やサービス、例えばプライベート・クラウドのIBM PureSystemsやパブリック・クラウドのIBM SmarterCloud Enterprise（以下、SCE）、SCE plusのイメージはDistributed Management Task Force（DMTF）のOpen Virtualization Format（以下、OVF）と呼ばれる仮想マシンのパッケージングのためのオープン・スタンダードに準拠しています。**図1**にSCEのイメージの例を示します。RAM.zipはクラウド環境に固有な管理情報で、プロビジョニング時にユーザーからのインプットを受け取ったり、インスタンスの起動スクリプトを指定してプロビジョニング後に初めて確定する情報（例えば仮想マシンのMACアドレス）を構成に反映することができます。

このイメージの開発効率を高め、クラウド・プラットフォーム間でのポータビリティーを高めるにはどうしたらよいでしょうか。まず開発効率ですが、イメージはOS、ミドルウェア、アプリケーション、それらに関する各種設定情報を含むため、非常に数が多くなる傾向があります。例えば3種類

の OS があり、OS ごとに 4 種類のミドルウェアと 5 種類のアプリケーション・プログラムがあるとします。説明を簡単にするため、ミドルウェアとアプリケーションに依存性はなく、すべてが共存可能とすると、これらの組み合わせは $3\times2^4\times2^5=1536$ 通りと非常に大きな数になります。これはイメージのスプロール現象と呼ばれる問題ですが、構成要素をコンポーネント化することで解決可能です。上記の例では 3 つの OS、12 のミドルウェア、15 のアプリケーションとして組み合わせ前の状態で管理するのです。これによって管理対象数は掛け算から足し算（3 + 12 + 15）に変わり 30 個に激減します。さらに各コンポーネントはそれぞれ OS、ミドルウェア、アプリケーションだけになり、それぞれの専門家による分業が可能になり、要員確保も容易になります。良いことばかりですが、最終的にコンポーネントを 1 つのイメージにまとめる必要があり、それが簡単にできなければ意味がありません。コンポーネント化された OS、そのほかのソフトウェアは、それぞれベース OS イメージ、SW バンドルと呼ばれ、基本的には**図 1** で示したイメージと同じ形をしています。これらを 1 つにするためにはバイナリー・データおよび管理情報をそれぞれ合体させることが必要ですが、その様子を**図 2** に示します。バイナリー・データの統合はイメージ統合時に行う場合とプロビジョニング後に起動スクリプトから行う場合があり、前者であれば一旦ベース OS イメージからインスタンスを作成（プロビジョニング）し、それに SW バンドルを導入した後のインスタンスをイメージに取り込みます。管理情報は XML ファイルなのでテキスト・エディターを使って手作業で合成することが可能です。しかしこれらは煩雑な作業のため、IBM では Image Construction and Composition Tool（以下、ICCT）を提供して簡単に統合できるようにしています。質のよい SW バンドルをそろえれば、ICCT を利用した効率的なイメージ開発が可能になります。

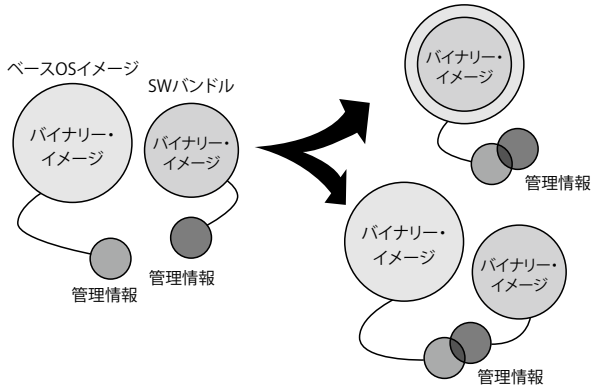


図2. イメージの合成

IaaS のポータビリティーについては、どこをインターフェースとしてポータビリティーを実現するかで戦略が分かれます。**図 3** の上段は VM の OS を標準インターフェースとして SW のポータビリティーを実現しています。この場合のポイントは、オープン・スタンダードもしくはデファクト・スタンダードとして一般に広く受け入れられている OS を採用することです。そして Java EE に準拠するプログラムを開発し、プログラムの中ではクラウド・プラットフォーム固有の API は使わないようにします。IaaS 固有のユーティリティー・サービスは使わず、代わりに VM の機能としてくくり出しておきます。こうすることで OS、Java EE が提供する標準インターフェースを介して Web アプリケーションをポータブルにすることができます。これは原始的な方法に見えますが、既存の IT を最大限活用し、使い慣れたアプリケーション・フレームワークも駆使し、広い範囲の IaaS に移植可能なプログラムを開発するための戦略として有効です。

もう 1 つは**図 3** の下段に示す、クラウド・プラットフォームを標準インターフェースとしてイメージのポータビリティーを実現する方法です。こちらの方がクラウド的な感じがしますが、実は簡単ではありません。IaaS はハイパーバイザーやクラウド管理レイヤーの仕組みが IaaS ごとに異なるため、たとえイメージが OVF に準拠していたとしても一般に可搬性がないからです。さらにハイパーバイ

```
[idcuser@vhost2456 image]$ ls -l /data/image
total 3714548
-rwxr-xr-x. 1 root root        713 Apr 22 09:31 BSS.zip
-rwxr-xr-x. 1 root root 4852179968 Apr 22 09:36 R7dTA2vtSVWTG8x0cr1SlQ.img
-rwxr-xr-x. 1 root root        196 Apr 22 09:37 R7dTA2vtSVWTG8x0cr1SlQ.mf
-rwxr-xr-x. 1 root root       2013 Apr 22 09:30 R7dTA2vtSVWTG8x0cr1SlQ.ovf
-rwxr-xr-x. 1 root root      82397 Apr 22 09:31 RAM.zip
-rwxr-xr-x. 1 root root      20648 Apr 22 09:31 Terms.zip
[idcuser@vhost2456 image]$
```

名前	種類	圧縮サイズ	パスワー...	サイズ
activation_scripts	ファイル フォルダー			
buildID	ファイル	1 KB	無	1 KB
GettingStarted.html	Firefox HTML Document	1 KB	無	2 KB
manifest.rmd	RMD ファイル	2 KB	無	5 KB
parameters.xml	XML ドキュメント	1 KB	無	1 KB
Rhel6.1-x64.bmp	ビットマップ イメージ	8 KB	無	451 KB
rhel62-x64.topology	TOPOLOGY ファイル	1 KB	無	5 KB
rhel62-x64.topologyv	TOPOLOGYV ファイル	2 KB	無	6 KB
TERMS.pdf	Adobe Acrobat Docum...	44 KB	無	48 KB

図1. イメージの実体はOVFパッケージ

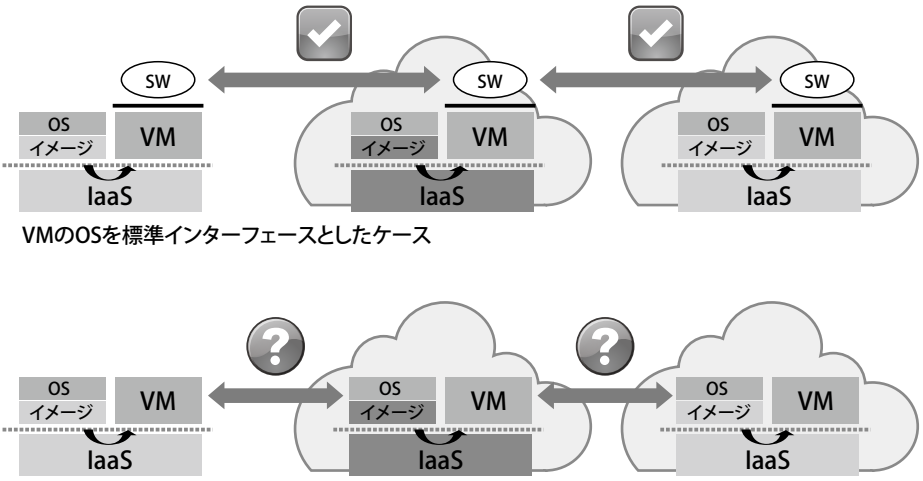


図3. IaaSのポータビリティー

ザーが同一であってもポータビリティーがあるとは一般にいえません。それは**図 1** で説明した管理情報の RAM.zip がクラウド・プラットフォームごとに異なるためです。ある IaaS 用に準備された起動スクリプトが別の IaaS では起動しないなどの問題が発生するのです。それではイメージによるポータビリティーは絶望的かといえばそうではありません。**図 2** で説明した SW バンドルはクラウド・プラットフォームではなく OS に依存するため、IBM クラウド間であればこの単位でポータブルなデザインが考えられます。例えばクラウド・プラットフォーム X のベース OS イメージと SW バンドルを 1 つのイメージにまとめてクラウド・プラットフォーム Y に移すことができなくても、クラウド・プラットフォーム Y のベース OS イメージと X からインポートした SW バンドルを組み合わせることは可能な場合があります。このように SW バンドルは開発効率、イメージの管理性、ポータビリティーのすべてに関係する重要なテクノロジーです。**図 3** について整理すると、上段はプロビジョニング後の標準的な VM に Java EE 準拠の SW を導入あるいはデプロイすることでポータビリティーが実現できることを、下段は SW を導入済みの同じイメージを異なるクラウドに導入しようとしても成功する保証はないことを示しています。ただし下段の場合でも SW バンドルを分離すればその部分についてはポータブルにできる可能性があります。

❸ サービス事業を支える PaaS (Platform as a Service) 技術 [3]

PaaS はアプリケーションの開発および実行プラットフォームを提供するサービスですが、その考え方は定義を含めてさまざまであり、クラウドのサービス・レイヤーの中では最もダイナミックに進化しています。IBM は SmarterCloud Application Services（以下、SCAS）として次の 5 つの機能領域を定めています。

・アプリケーション・ライフサイクル
・アプリケーション・リソース
・アプリケーション環境
・アプリケーション管理
・インテグレーション

本稿では SCE でパイロット・プログラムを展開中（本稿執筆時点）のアプリケーション環境（Application Environments）を中心に説明します。

ここまで 1 つの VM についてのみ考えてきましたが、実際のサービスでは複数のサーバー（VM）を組み合せて実現することが多くあります。2 つ以上の VM があるということは VM 間の相互接続関係、すなわちトポロジーがあるということです。SCAS のアプリケーション管理は仮想パターンによりこのトポロジーを定義しており、仮想パターンには仮想システム・パターン（Virtual System Pattern）、仮想アプリケーション・パターン（Virtual Application Pattern）の 2 種類があります。

仮想システム・パターンは、パート（Parts）、スクリプト・パッケージ（Script Packages）、アドオン（Add-ons）からなる定義体で、これを PaaS の実行環境にデプロイすると仮想システム・インスタンス（Virtual System Instance）、すなわち複数 VM から成るサービスになります。 パートは IaaS で説明したイメージにメタデータを追加したもので、VM に対応しています。スクリプト・パッケージは VM に導入されるスクリプト、アドオンはストレージや vNIC（Virtual Network Interface Card）など VM のフィーチャーの指定するもので、どちらも仮想システム・パターン内のパートに従属します。少し複雑ですが、簡単に言えば複数のパート（その元はイメージ）の関係をトポロジーとして定義しているのが仮想システム・パターンです。パートの配置は専用のパターン・エディターで視覚的に行えますが、特に IBM が提供するパートはあ

らかじめパート間の関係が定義されており、例えばIBM WebSphere Application Serverのパートと IBM DB2のパートをエディターのカンバスに置くと自動的に両者が矢印でつながります。このように簡単に仮想システム・パターンが作れるようになっています。

仮想アプリケーション・パターンは、コンポーネント（Components）、リンク（Links）、ポリシー（Policies）からなる定義体で、これをPaaSの実行環境にデプロイすると仮想アプリケーション、すなわち複数VMからなるサービスになります。コンポーネントとはパターンを構成する部品で、アプリケーション・コンポーネント、データベース・コンポーネント、ユーザー・レジストリー・コンポーネントなど多様なカテゴリーのどれかに属します。コンポーネントはミドルウェア、もしくは外部のミドルウェアと接続するコネクターに対応すると考えると分かりやすいです。リンクはコンポーネント同士の接続、ポリシーはコンポーネントやパターン全体の品質を定義します。仮想アプリケーション・パターンはひな形であるパターン・タイプ（Pattern Types）から構成され、さらにパターン・タイプは複数のプラグイン（Plug-ins）を含んでおり、これらをカスタマイズすることで仮想アプリケーション・パターン自体をカスタマイズすることが可能です。しかし仮想アプリケーション・パターン利用の真髄は、IBM が提供する、もしくは IBM のビジネス・パートナーから提供されるベスト・プラクティスに基づくパターン、例えばWebアプリケーション・パターン（Web Application Pattern）をそのまま利用し、仮想アプリケーション・ビルダー（Virtual Application Builder）を使って標準パターンの範囲内でアプリケーション・モデルを完成させデプロイすることにあります。アプリケーション開発エンジニアは開発したプログラムをWAR（Web Archive）、EAR（Enterprise Archive）、EBA（Enterprise Bundle Archive）の形にパッケージしてアプリケーション・コンポーネントに含めることができます。またポリシーを指定して、VMのリソース利用やレスポンス時間によって自動的にアプリケーションをスケールアウトすることができるので、アプリケーション開発者は開発業務に専念し、アプリケーションとデータベース間の接続やアプリケーションの負荷分散にかかわるミドルウェアの設定作業から解放されることになります。仮想システム・パターンでは利用者がVMのイメージを作成し、それを配置しましたが、仮想アプリケーション・パターンではアプリケーション・パッケージを作成し、それを配置しています。前者は自由度が高く、後者は利用者にインフラやミドルウェア構築スキルがほとんど要求されず、高い抽象度を持つことが特徴です。

最後にPaaSのポータビリティーについて考えます。**図4**の上段で示すように、一般的に異なるクラウド・ベンダーのPaaS間にポータビリティーはありません。しかしIBMのPaaSは異なるデリバリー・モデル、すなわちパブリック・クラウドとプライベート・クラウドで同じPaaSを実現しているため、パブリック・クラウド上で開発・テストした仮想アプリケーション・パターンをそのままプライベート・クラウドに転送・デプロイして実行・運用することができます。これはIBM PaaSのアーキテクチャーがインフラと疎結合で、IaaS上のサービス・インスタンスとして実装可能なためです。仮想システム・パターンはすでに見たようにイメージのクラウド・プラットフォームへの依存性のためICCTによるイメージの再構成が必要ですが、仮想アプリケーション・パターンではそのようなことも必要ありません。**図4**の下段はIBM PaaSが異なるクラウド・デリバリー・モデル間で仮想アプリケーション・パターンのポータビリティーを実現していることを示しています。

図5はSCEにおけるPaaSの実装を示しています。仮想アプリケーション・ビルダーやPaaS実行環境はIBMの標準アプライアンス製品であるIBM Workload Deployer（以下、IWD）に実装されています。利用者はまずIWDをSCEのサービス・インスタンスとしてプロビジョニン

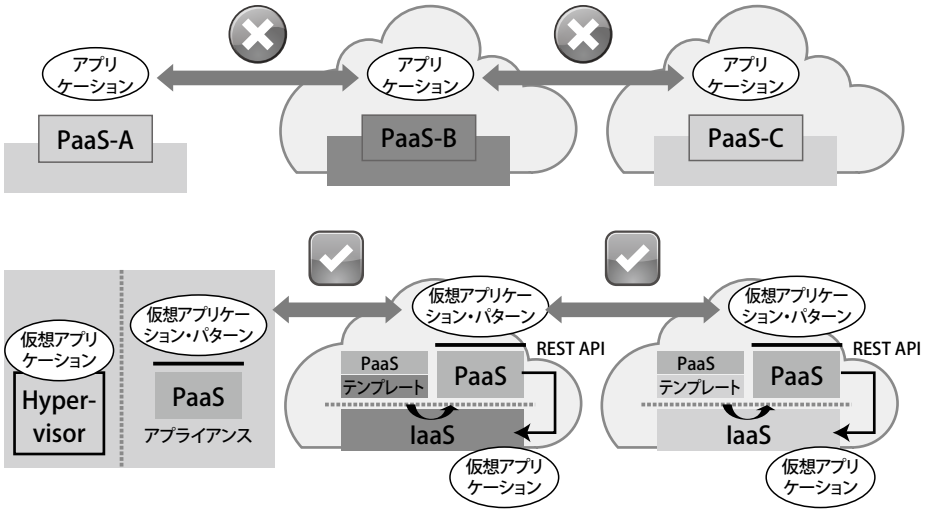
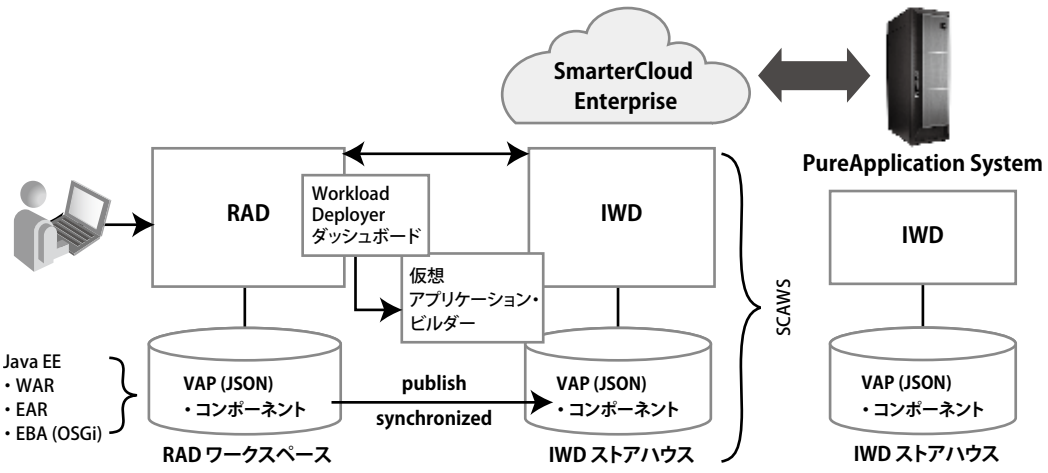


図4. PaaSのポータビリティーについて

グし、それをPaaSとして利用します。SCEのサービス・インスタンスは汎用性の高い機能を備え、サービス提供者はサービス・ディスクリプターを作成することで複数VMから成るアプリケーションをSCE上にプロビジョニングすることができます。サービス・ディスクリプターでは1つ1つのVMやストレージなどのリソースをXMLで定義し、またそれらをプロビジョニングするJavaScriptを記述します。ユーザーがセルフサービス・ポータルからサービスを選ぶと、サービス・ディスクリプターで指定したJavaScriptがWebブラウザーにダウンロードされ、これがSCEのREST APIを呼び出すことで次々とVMがプロビジョニングされます。拡張性の高い仕組みですがSCEでしか使えず、サービス・ディスクリプターにポータビリティーはありません。そこでクラウド・インフラに依存しないIWDをまずサービス・インスタンスとしてプロビジョニングし、ユーザーには仮想パターンによるPaaSを提供します。筆者はこれを「PaaSのブート・ストラップ」と呼んでいます。IWDは仮想パターンをインスタンスに変換する際にSCE APIをコールすることでSCE上にVMをプロビジョニングします。このサービスはSCAWS（SmarterCloud Application Workload Service）と呼ばれ、SCASのアプリケーション環境機能を提供します。アプリケーション開発者はRational Application Developer（以下、RAD）が導入されたPC（Eclipse PC）で開発を行い、開発したアプリケーションを仮想アプリケーション・パターンで包んでSCAWSへ送ります。**図5**から分かるように仮想アプリケーション・パターンはJSON（JavaScript Object Notation）ファイルとして実装されており、トポロジーやポリシーをコード化することで管理性を高め、自動化を実現しています。RADからSCASへ、このJSONファイルと複数のコンポーネント（EARファイルなど）が送られます。この後、RADではプログラムの変更・開発を続け、SCASでは仮想アプリケーション・ビルダーにより仮想アプリケーション・パターンの修正を続けることができます。RADとSCASは相互に変更内容の同期を取り一貫性を維持し続けます。完成した仮想アプリケーション・パターンはSCAS上でデプロイされます。これによりパターンは仮想アプリケーション（インスタンス）となり、SCAS上で実行されます。

図5の右側はPaaS環境をPureApplication Systemに切り替えた例です。どちらも同じIWDによるPaaS環境であり、またRADには最新の仮想アプリケーション・パターンがシンクロナイズされているため、RADの接続先をPureApplicaiton Systemに切り替えるだけで仮想アプリケーションがプライベート・クラウドに転送・デプロイされて実行されます。開発チームがSCE上でファンクション・テストまで行ったまったく同じ内容と構成を、手作業を介さずに本番運用チームのPureApplication System環境に引き継ぐことができ、開発・運用の連携を促し、サービスの継続的改善が促進されることが期待されます［4］。

RAD PCは協働開発環境の一員としてIBM Rational Team Concert（以下、RTC）のプロジェクト・メンバーになることができます。**図6**に示すように、このRTCもSCEのサービス・インスタンスとして提供され、これに要求管理サービス、品質管理サービスを加えたものがアプリケーション・ライフサイクル（Application Lifecycle）機能です。これにより従来のVMのライフサイクル管理（イメージ→プロビジョニング→運用・監視→ディプロビジョニング）とアプリケーション・ライフサイクル管理（要件定義→デザイン→コーディング→ソースコード管理→単体テスト→ビルド→パッケージング→ファンクション・テスト→デプロイ）が統合され、サービスをスピーディー



IWD：IBM Workload Deployer　RAD：Rational Application Developer
VAP：Virtual Application Pattern　JSON：JavaScript Object Notation

図5. SCEにおけるPaaSの実装

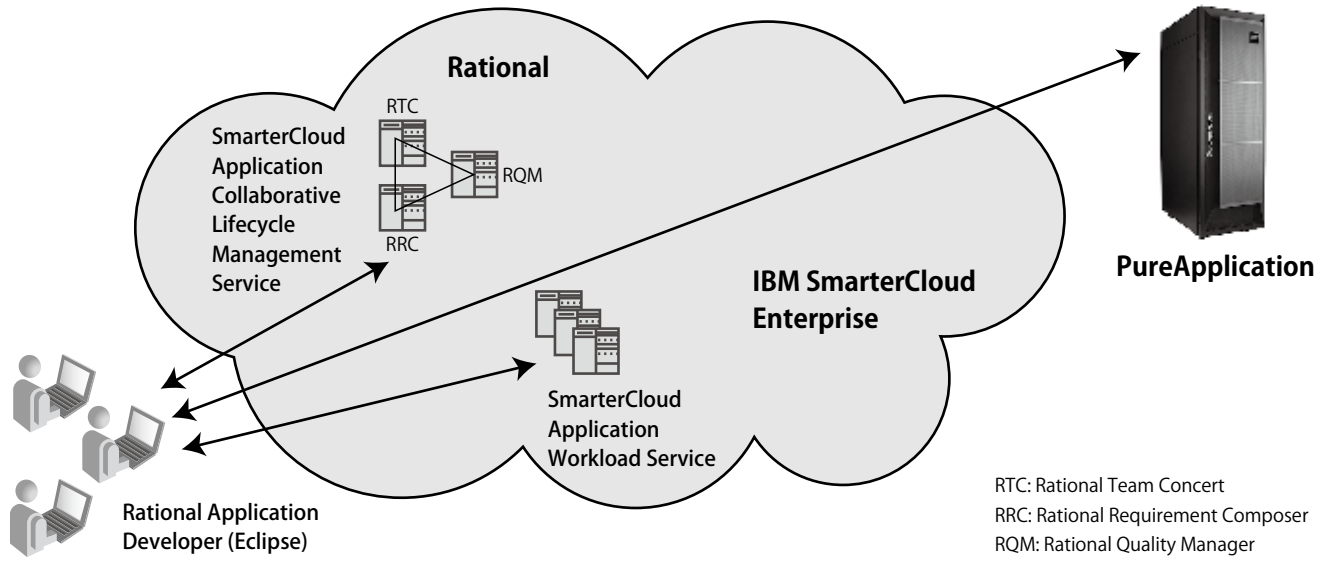


図6. PaaS上のアプリケーション・ライフサイクル・マネジメント

かつ継続的に改善することが可能になります［5］［6］。

❹ まとめ

サービス事業を支えるクラウド・コンピューティングについて、イメージや仮想パターンの開発、ポータビリティーを中心に説明してきました。ベース・イメージとSWバンドル、IaaSとPaaS、仮想パターンとプログラム・パッケージなど、「分離と疎結合」は変化の時代のキーワードです。チーム開発、アジャイル開発、継続的インテグレーション & デプロイを支援するアプリケーション・ライフサイクル管理においても多様な開発・運用ツールの疎結合が不可欠で、Linked Dataによるヘテロジニアスな連携が進められています［7］。これらにサーバーおよびクライアント（モバイルを含む）双方で進化を続けるアプリケーション・フレームワークを加えると新サービス時代の主要なパズルのピースが見えてきます。Java EE準拠のアプリケーション・フレームワークから生成されるEARファイルは仮想アプリケーション・パターンに載せられる点に注目してください。IBMはこれらすべての領域でお客様のイノベーションをご支援していきます。

［参考文献］

［1］ 紫関昭光：IBM Smart Business Cloud Enterprise 活用ガイド ～導入から構築、運用まで エンタープライズ・クラウドのすべて～ volume.1，廣済堂（2012）.

［2］ Distributed Management Task Force: Open Virtual Format Specification v1.1, http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_1.1.0.pdf

［3］ S.Imazeki et al.: IBM Workload Deployer: Pattern-based Application and Middleware Deployments in a Private Cloud, IBM Redbook (SG24-8011-00), http://www.redbooks.ibm.com/redbooks.nsf/RedbookAbstracts/sg248011.html

［4］ C.Brealey et al.: Preparing for IBM PureApplication System, IBM developerWorks, http://www.ibm.com/developerworks/websphere/library/techarticles/1204_abrams/1204_abrams.html

［5］ S.Krishna and TC Fenstermaker: IBM Rational Team Concert 2 Essentials, Packt Publishing (2011).

［6］ P.Duvall et al.：継続的インテグレーション入門 - 開発プロセスを自動化する47の作法，日経BP社（2009）.

［7］ T.Heath and C.Bizer: Linked Data: Evolving the Web into a Global Data Space, Morgan & Claypool Publishers, http://www.morganclaypool.com/doi/abs/10.2200/S00334ED1V01Y201102WBE001

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
スマーター・クラウド事業部
理事　クラウド・マイスター

紫関 昭光 **Akimitsu Shiseki**

［プロフィール］

1981年日本IBM入社。藤沢研究所、大和研究所にて通信機器開発、中型システム開発、ソフトウェア開発に従事。IBM Asia Pacific クロスインダストリーにてビジネス・インテリジェンス・マーケティングに従事。開発製造にてビジネス・ディベロップメント理事。2007年よりグローバル・テクノロジー・サービス事業ITSデリバリー、クラウド事業を経て現職に至る。