レガシー・アプリケーションをコンテナに移行する

はじめに

多くの組織が初期のパブリッククラウド・プロジェクトを成功させました。しかし、これらは基本的に、パブリッククラウドで実行するのに適した候補として慎重に選ばれた、新しいグリーンフィールド・アプリケーション・プロジェクトです。このような成功の結果、IT 組織はクラウド・コンピューティングが提供する弾力性、スケーラビリティ、およびデプロイメントの速度の魅力を知ることになりました。クラウドテクノロジーを利用することで、IT 組織は開発者やビジネス部門の要求に対してより迅速に対応できるようになります。

レガシー・アプリケーションは通常、セキュリティ、規制、データの局所性、パフォーマンスなどの懸念からパブリッククラウドのデプロイには考慮されません。レガシー・アプリケーションの多くはクラウド・コンピューティング以前に開発されたものであるため、既存のインフラストラクチャにデプロイしたままの方がシンプルだと思われているかもしれません。しかしこの考えは、モダナイゼーションを進めようとする組織にとってボトルネックになりかねません。レガシー・アプリケーションの継続的な稼働にかかるコストが IT コストの大半を占めることが多いため、レガシー・アプリケーションへの対応なくして、コスト削減を図りながら応答性を高めることはできません。

コンテナはパブリッククラウド・プロバイダーが提供するサービスの多くを可能にする重要なテクノロジーです。コンテナの設計は自動化に多くの可能性をもたらします。コンテナは、クラウドのような自動化を提供するプラットフォームと組み合わせることで、アプリケーションを実行するための魅力的な環境となります。レガシー・アプリケーションをコンテナに移行することで、モダナイゼーションの障壁の多くを取り除くことができます。

レガシー・アプリケーションをコンテナに移行する理由

スケーリングと迅速な対応の必要性

レガシー・アプリケーションは多くの場合、固定された限られたリソースのインフラストラクチャにデプロイされています。リソース使用率が低いことも珍しくありません。そして需要が増加した場合、長いリードタイムとコストを費やすことなく規模を拡大することは困難です。パブリッククラウドで稼働する SaaS (Software-as-a-Service) アプリケーションの成功によって、応答性とコストに対するユーザーとビジネスの期待は変化しました。社内のアプリケーションが素早く進化できない理由の説明は難しいものになる可能性があります。

レガシー・アプリケーションの多くはこれまで安定的かつ予測可能な成長を遂げてきました。しかし、ユーザー主導の新たな需要の台頭により、レガシー・アプリケーションで利用可能なリソースを迅速に拡張する必要性が生じています。ユーザー主導の需要を IT 組織が予測することは、以下の理由から困難です。

- モバイルアプリケーションやコネクテッド・アプリケーションでは、既存のアプリケーションへのアプリケーション・プログラミング・インタフェース (API) レベルのアクセスを必要とするのが一般的になっている。
- データサイエンスと機械学習の台頭により、データアクセスに対する需要がさらに高まっている。
- データや API を利用するアプリケーションと同様に、需要の一部は IT 組織の外部にある。

成長を予測し、需要をコントロールすることは難しいため、組織が迅速に対応できるように既存のアプリケーションを再配置する必要があります。先進的なクラウド規模のアプリケーションは、需要に応じて実行中のコンテナ数を増減させ、アプリケーションの容量を増減させるプラットフォーム上のコンテナで実行することでこの課題に対処しています。

コンテナでレガシー・アプリケーションを実行するメリット

- 可搬性：アプリケーションをインフラストラクチャから分離し、コンテナをサポートする任意のプラットフォームで実行する能力。
- スケーラビリティ：リソースをより適切に利用しながら、需要に対応するために必要に応じてスケールアップ (またはスケールダウン) する能力。
- 柔軟性：無駄にリソースを拘束することなく、必要なときにコンテナをデプロイしてテスト環境を作成する容易性。
- 言語とテクノロジーの多様性：数十年前のコードか新しく書かれたコードかを問わず、レガシー・テクノロジーと先進的なテクノロジーの共存を可能にするための言語、データベース、フレームワーク、ツールの選択肢のサポート。

変化を可能にするためのレガシーシステムの位置付け

通常、レガシーシステムと新しいグリーンフィールド開発の機会は繋がっています。一般的に、新しいアプリケーションやサービスはレガシー・アプリケーションからのデータを必要としたり、レガシーシステムでトランザクションを実行することでサービスを実行したりします。新しいテクノロジーで実装された新しいインタフェースやサービスを、レガシーシステムの前に置くというのがモダナイゼーションの一般的なアプローチです。

パブリッククラウド上で動作する新しく開発されたアプリケーションと社内で実行されているレガシー・アプリケーションを繋げると、複雑なセキュリティ上の課題が発生します。特にネットワーク関連の問題は、追跡や診断が困難です。この問題は、先進的なツールが利用できない古いインフラストラクチャ上でレガシー・アプリケーションが実行されている場合、より困難となります。

レガシーシステムに依存する新しいアプリケーションにはテストが必要です。先進的な開発手法では、品質と信頼性を向上させるために自動化されたテストを使用する傾向がありますが、レガシー・アプリケーションのテスト環境にはより多くのリソースが必要になるでしょう。また、開発チームは新しいコードを開発してテストするために、追加の (場合によっては分離された) レガシー・アプリケーション用テスト環境にアクセスすることが必要になる場合もあります。

レガシー・アプリケーションをコンテナでデプロイすることで、変化に対する障壁を取り除き、進化に必要な柔軟性を獲得できます。このプロセスは、古いインフラストラクチャからアプリケーションを切り離すことから始まります。そして同じプラットフォームを使ってレガシー・アプリケーションと新しいグリーンフィールド開発をホストします。どちらも同じコンテナやクラウドプラットフォーム上で共存でき、同じツールで管理できます。古いインフラストラクチャの制約を受けずに、レガシー・アプリケーションに自動化ツールや先進的な管理ツールが使用されるようになれば、運用効率は向上します。

レガシー・アプリケーションをコンテナに移行する際の考慮事項

永続ストレージ

クラウドネイティブではないアプリケーションは、データ、ログ、場合によっては構成用の永続ストレージを必要とします。しかし、コンテナは短期間しか存在しないことを前提に設計されています。特別に手配しない限り、コンテナ内部に書き込まれたものは、コンテナが再起動されると失われます。コンテナが永続ストレージにアクセスできるようにすることで、レガシー・アプリケーションを格納できます。コンテナは通常、複数のマシンで構成されるクラスタ上で実行されるため、永続データ用のストレージは、コンテナが実行可能なクラスタ内のすべてのマシンで利用できる必要があります。利用可能なストレージの種類は、コンテナ・プラットフォームとそれを実行するインフラストラクチャに大きく依存します。

コンテナ・オーケストレーション

ほとんどのアプリケーションは、同時に実行し、互いに接続する必要があるコンテナで構成されています。例えば、3 層構造のアプリケーションの各層を構成するコンポーネントは別々のコンテナで実行されます。Web やアプリのコンテナは、需要の増加に応じてクラスタ内のより多くのマシンに動的にスケールアウトできる機能によるメリットが得られます。コンテナのスケジューリングおよび管理のプロセスはコンテナ・オーケストレーションと呼ばれており、コンテナ・プラットフォームの重要な役割となっています。

ネットワーク

多くの場合、アプリケーションには特定のネットワーキング要件があり、それがデプロイ方法を決める鍵となります。コンテナ環境で仮想ネットワークを再作成する必要がある場合や、場合によっては、物理的なネットワーキング・ハードウェアをコンテナ環境で仮想化する必要があるかもしれません。ストレージと同様に、アプリケーション用の仮想ネットワークは、コンテナが実行されるすべてのホストで利用可能でなくてはなりません。コンテナ・プラットフォームは、異なるコンテナで実行されるアプリケーションのコンポーネントを接続する仮想ネットワーク環境を管理し、それらのコンポーネントをコンテナ・プラットフォーム上で実行される他のアプリケーションから分離します。

Kubernetes について

Kubernetes は、コンテナ・プラットフォームのデファクトスタンダードになっています。これは、大量のコンテナを大規模に実行している Google の経験に基づいて、コンテナのデプロイと管理を自動化するオープンソース・プラットフォームです。Kubernetes は、アプリケーションの望ましい最終状態が維持されるようにするために、コンテナの自動再起動、別のホストでのコンテナの再スケジュール、自動スケーリングのようなユースケース向けのコンテナの複製など、自動化された自己修復メカニズムを使用します。Kubernetes は、一般的な Docker コンテナ形式を含む Linux コンテナでネイティブに動作します。

Red Hat は Kubernetes プロジェクトにおいて Google に次いで 2 番目に大きく貢献するコントリビューターです。[1] 2015 年以降、Kubernetes プロジェクトは Cloud Native Computing Foundation が管理しています。そのオープン性によって、Kubernetes は業界で広く導入され、急速なイノベーションを加速させています。また、Kubernetes 上に構築されるオープンソース・プロジェクトも増加しています。

アプリケーション用コンテナの構築

開発者には、アプリケーションと必要な依存関係をコンテナイメージに構築するためのツールが必要です。このプロセスは、コード変更や完成リリースのたびに繰り返す必要があります。ロールアウトの際、運用担当者や開発者には、現在実行中のコンテナイメージと入れ替える新しいイメージをデプロイする機能も必要です。これらのタスクを実行するための低レベルのツールは既に存在しますが、コンテナ・プラットフォームがあればこのプロセスははるかに容易になります。

アプリケーションを実行するためにコンテナを構築するには、通常、アプリケーションを実行するための言語、ランタイム、フレームワーク、アプリケーション・サーバーが必要になります。これらは、ベースコンテナイメージを基盤として、ビルドプロセス中に取り込むことができます。ベースイメージには多くのソースがありますが、定評のある信頼できるソースからそれらを取得することが課題になっています。ベースイメージは安全、最新、かつ既知の脆弱性がないものでなければなりません。脆弱性が発見された場合、ベースイメージは更新されなければなりません。また、コンテナが古いイメージに基づいているかどうかをユーザーが調べる方法も必要です。

パブリッククラウドの課題

IT 組織がパブリッククラウドを導入する際に直面する課題の 1 つとして、パブリッククラウドが提供するインフラストラクチャ、管理、自動化ソフトウェアが、IT 組織が自社のデータセンターで使用しているものとは異なるという点が挙げられます。パブリッククラウドのツールやサービスの多くはオンプレミスで利用できないため、社内で実行されているアプリケーションには使えません。

多くの企業は、地理的な可用性、多様性、コストなどの理由から、複数のパブリッククラウドを使用することを選択します。しかし、各パブリッククラウドプロバイダーは、ベンダー固有のインタフェース、ツール、サービスを提供しています。

コンテナとクラウドは、自動化によって運用効率を向上させる膨大な可能性を秘めています。コンテナは、DevOps 手法と文化を実装するための理想的な環境です。しかし、ホストされたアプリケーションがあるあらゆる場所で異なるプラットフォームを使用するクラウド戦略では、把握し学ぶべきことが多すぎるため、運用担当者や開発者が過負荷になる場合があります。

Red Hat のアプローチ：クラウド体験をあらゆる場所で

Red Hat® OpenShift® はエンタープライズ対応の Kubernetes コンテナプラットフォームです。ハイブリッドクラウドやマルチクラウドのデプロイメントを管理するフルスタックの自動運用機能を備えており、パブリッククラウドのシンプルさと自動化を提供します。Red Hat OpenShift には、エンタープライズグレードの Linux® オペレーティングシステムと、コンテナランタイム、ネットワーキング、モニタリング、レジストリ、認証、承認のソリューションが含まれています。

Red Hat OpenShift Container Platform は、オンプレミスのデータセンターでもプライベートクラウドでも、任意のインフラストラクチャにデプロイできます。インフラストラクチャを自社で管理したくない場合、ほとんどのパブリッククラウドプロバイダーがマネージドサービスとして Red Hat OpenShift を提供しています。

一貫したハイブリッドクラウド基盤で運用を効率化

Red Hat OpenShift は、新しい開発がクラウドプラットフォーム上で行われる中、レガシー・アプリケーションがオンプレミスに留まる必要がある場合に発生する課題に対処するのに役立ちます。Red Hat OpenShift は基盤となるクラウドやコンテナ・プラットフォームの詳細を抽象化することで共通のアプリケーション・プラットフォームを構築し、ハイブリッドやマルチクラウドへの移行を容易にします。

Red Hat OpenShift は新旧アプリケーションで共通の運用インタフェースを提供するので、内部で実行されるか外部で実行されるかを問わず、運用を効率化できます。アプリケーションの実行場所に関係なく、同じツール、コンソール、手順を使用できます。学習が容易になり、運用担当者はより早く生産性を上げることができます。複数の環境での操作方法をすべて把握しておく必要がなくなるため、より迅速に問題を診断し、解決できるようになります。

1 Stackalytics、「Kubernetes」、2019 年 12 月 6 日にアクセス。https://www.stackalytics.com/cncf?module=kubernetes

共通のアプリケーション・プラットフォームにより、アプリケーションの可搬性とデプロイメントの柔軟性が向上します。コンテナには、複数のコンテナをオーケストレーションして完全なアプリケーションを提供するために必要なデプロイメントの全詳細が含まれているわけではありません。Kubernetes ではデプロイと構成の詳細が複数の YAML ファイルで保存されています。Red Hat OpenShift が Kubernetes よりも高い付加価値を提供する領域の 1 つとして、グラフィカル・ユーザー・インタフェース (GUI) とデプロイメント・テンプレートを提供することにより、運用担当者や開発者が YAML ファイルを手作業で編集する必要性がなくなるという点が挙げられます。

デプロイメント・テンプレートは、Red Hat OpenShift でのアプリケーションのデプロイと、ある OpenShift クラスタから別のクラスタへのアプリケーションの移動のプロセスを効率化します。テンプレートはアプリケーション・コードの一部とすることも、個別に保管することも可能です。アプリケーションは Red Hat OpenShift サービスカタログに追加でき、アプリケーションとソフトウェアコンポーネントはマウスのクリック操作でデプロイできます。

複数の Red Hat OpenShift クラスタを管理できるよう、Red Hat OpenShift 4 はで統合ハイブリッドクラウド・コンソールが導入されました。この機能は、オンプレミスまたは複数のクラウド上で実行可能なクラスタ全体で、一元管理および可視化ツールを提供します。

コンテナでのアプリケーション開発

アプリケーションをコンテナに移行するには、まずアプリケーションコードをコンテナイメージにビルドする必要があります。Red Hat OpenShift は、開発者がリソースのプロビジョニングを待つことなくコンテナを構築して実行できるセルフサービス・プラットフォームを提供します。これは、Red Hat OpenShift が Kubernetes よりも高い付加価値を提供する主要領域の 1 つです。

Red Hat OpenShift を通じて、開発者は継続的インテグレーション/継続的デリバリー (CI/CD) のための自動ビルドを設定できます。新しいコードがソースコードのバージョン管理システムにチェックインされるたびに、ビルドを自動的にトリガーできます。ビルドが正常に完了すると、以前のバージョンの代わりに新しいバージョンを自動的にデプロイできます。この機能は自動テストと継続的改善に役立ちます。Red Hat OpenShift には、洗練された自動ビルドパイプラインを作成するための豊富な機能があります。開発者は Jenkins などの使い慣れたツールを使用でき、ビルド環境をゼロから構築するような複雑な作業は必要ありません。

IT 運用担当者による制御体制はそのままに、開発者はクラスタへの管理アクセスなしで作業できます。Red Hat OpenShift は、セキュリティを備えたマルチテナントをサポートしています。ビルドの実行でも、実行中のコードをデバッグするためのログインでも、開発者が行うタスクはすべて Red Hat OpenShift 上のコンテナ内で実行されます。開発タスクはコンテナ内で実行されるため、他のコンテナやクラスタ自体から分離されます。

開発者向けのツール

Red Hat は、開発者がコンテナで実行するアプリケーションを構築するのに役立つツールを数多く提供しています。

- Red Hat CodeReady Studio は、コンテナおよび複数のプログラミングモデル向けの広範なツールセットを備えた従来型のデスクトップ統合開発環境 (IDE) です。
- Red Hat CodeReady Workspaces は、Red Hat OpenShift 上で実行される Kubernetes ネイティブの開発者ワークスペースサーバーです。ブラウザーベースの IDE が提供されるため、開発者はソフトウェアをインストールしたり、ローカルマシンにコードをコピーしたりする必要がありません。
- Red Hat CodeReady Containers は事前構成済みの最小限の Red Hat OpenShift 環境です。開発者はノート PC で実行することができ、完全に自己完結型の開発環境を得ることができます。
- Red Hat Container Catalog は、開発者がベースイメージとして使用できる、信頼できるソースからのテスト済みコンテナイメージのライブラリを提供します。

- Red Hat OpenShift Application Runtimes は、クラウドネイティブ開発を単純化する、複数の言語とプログラミングスタイルを備えた Red Hat OpenShift の統合ランタイムのコレクションです。
- Red Hat Application Migration Toolkit は、開発者がレガシー・アプリケーションのコードを評価し、現在のアプリケーション・サーバーやミドルウェアなどの先進的プラットフォームで実行するために必要な変更を判断するのに役立つツールの集合体です。

レガシー・アプリケーションをコンテナに移行する

アプリケーションのコンテナを構築したら、アプリケーションをデプロイするための次なるステップは、ストレージとネットワークの設定です。永続ストレージのニーズに対応するため、Red Hat OpenShift で定義されたアプリケーションは永続ストレージボリュームを使用するように設定できます。これは、稼働時にアプリケーションのコンテナに自動的にアタッチされます。開発者は、運用によってプロビジョニングされたストレージプールから引き出し、コンテナベースのアプリケーション用の弾力性に優れたストレージを管理できます。Red Hat OpenShift Container Storage を使用すれば、ソフトウェア・デファインドの永続ストレージを作成できます。Red Hat OpenShift Container Storage は、Red Hat OpenShift クラスタ上で実行されるアプリケーションにブロックストレージ、ファイルストレージ、オブジェクトストレージのアクセス手法を提供します。

コンテナで実行されるアプリケーションのための仮想プライベート・ネットワーキング、ルーティング、ロードバランシングは、Kubernetes と Red Hat OpenShift が提供するプラットフォームの一部として組み込まれています。ネットワーキングは、アプリケーションのデプロイメント構成の一部として、宣言形式で指定されます。アプリケーション固有のネットワーク構成をソースコードと一緒に保存し、Infrastructure as Code にすることができます。アプリケーション固有のインフラストラクチャ構成を各アプリケーションに関連付けることで、アプリケーションのデプロイメントを移動、追加、変更する際の信頼性が向上します。

ソフトウェア・デファインドのルーティングとロードバランシングは、アプリケーションが自動的にスケールアップまたはスケールダウンできるようにする上で重要な役割を果たします。さらに、Red Hat OpenShift 上で実行しているアプリケーションは、ローリングデプロイメントを活用してリスクを低減できます。Red Hat OpenShift の組み込みのサービスルーティングでは、ローリングデプロイメントの戦略を使用して、ユーザー集団のサブセットで新しいコードをテストできます。何か問題が発生した場合、Red Hat OpenShift 上のコンテナで簡単に以前のバージョンにロールバックできます。

最後に、Red Hat OpenShift Service Mesh は分散アプリケーションのレジリエンシー (回復力) とパフォーマンスを向上させます。OpenShift Service Mesh はサービス間通信のロジックを抽象化して専用のインフラストラクチャ・レイヤーとするので、通信の効率と分散アプリケーションの回復力が高まります。OpenShift Service Mesh はセキュリティ重視のエンタープライズ・プラットフォームに Istio サービスメッシュ、Jaeger (トレース)、Kiali (可視性) を導入します。

アプリケーション環境の改善

レガシー・アプリケーションを Red Hat OpenShift 上のコンテナで実行すると、改善の機会が訪れます。CI/CD、ビルドとデプロイの自動化、自動テスト、ローリングデプロイメントを使用することで、新しいコードリリースをより高い信頼性でより頻繁に行うことができます。より頻繁にコードをリリースできるということは、変化するビジネス需要に対する対応力が高まるということです。

新しいテクノロジーで実装された新しいインタフェースやサービスを、レガシーシステムの前に置くというのがモダナイゼーションの一般的なアプローチです。すべてをコンテナで実行すると、各コンテナ内でどの言語やテクノロジーを実行しているかは関係なくなるので、このアプローチははるかに容易です。Red Hat OpenShift には仮想ネットワーキング機能とサービスメッシュがあるので、アプリケーション・コンポーネントを容易に高い信頼性で接続できます。

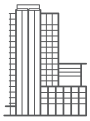
Red Hat OpenShift により、レガシー・アプリケーションとともに最新のミドルウェアをデプロイすることも容易になります。Red Hat は、コンテナ内の OpenShift クラスタ上で実行可能な統合およびメッセージングシステム、ビジネスプロセス管理、意思決定管理ソフトウェアを提供しています。これらを使用してアプリケーションを接続し、アジャイル・インテグレーションを実現できます。

まとめ

ハイブリッドクラウドとマルチクラウドに対する Red Hat のアプローチは、オンプレミスかパブリッククラウドかを問わず、新旧のアプリケーションに対応する共通のアプリケーション・プラットフォームを提供します。その結果、アプリケーションの可搬性が向上し、企業は柔軟に最も合理的な場所でワークロードを実行できます。基盤となる複数のクラウドやコンテナ・プラットフォームの詳細が抽象化されるため、アプリケーションをどこで実行するかに関係なく、運用担当者や開発者の生産性が向上します。

レガシー・アプリケーションをコンテナ化し、新旧のアプリケーションを Red Hat OpenShift 上で実行することには多くのメリットがあります。コンテナベースのアーキテクチャを Kubernetes および Red Hat OpenShift と連携させると、アプリケーションの信頼性とスケーラビリティが向上するとともに、開発者と運用担当者のオーバーヘッドが減少します。Red Hat OpenShift のフルスタック自動化、開発者のセルフサービス、CI/CD 機能は、継続的な改善プロセスの基盤にもなります。

コンテナの詳細およびコンテナの大規模な実行について詳しくは、こちらをご覧ください。
https://www.redhat.com/ja/solutions/hybrid-cloud-infrastructure#scale



Red Hat について

エンタープライズ・オープンソース・ソフトウェア・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 Linux、ハイブリッドクラウド、コンテナ、および Kubernetes テクノロジーを提供しています。Red Hat は、新規および既存 IT アプリケーションの統合、クラウドネイティブ・アプリケーションの開発、Red Hat が提供する業界トップレベルのオペレーティングシステムへの標準化、複雑な環境の自動化、セキュリティ保護、運用管理を支援します。受賞歴のあるサポート、トレーニング、コンサルティングサービスを提供する Red Hat は、フォーチュン 500 企業に信頼されるアドバイザーです。クラウドプロバイダー、システムインテグレーター、アプリケーションベンダー、お客様、オープンソース・コミュニティの戦略的パートナーとして、Red Hat はデジタル化が進む将来に備える企業を支援します。

アジア太平洋
+65 6490 4200
apac@redhat.com

オーストラリア
1 800 733 428

インド
+91 22 3987 8888

インドネシア
001 803 440 224

日本
03 4590 7472

韓国
080 708 0880

マレーシア
1 800 812 678

ニュージーランド
0800 450 503

シンガポール
800 448 1430

中国
800 810 2100

香港
800 901 222

台湾
0800 666 052

fb.com/RedHatJapan
twitter.com/RedHatJapan
linkedin.com/company/red-hat

jp.redhat.com
#F18866_0120