RED HAT OPENSHIFT

redhat.

ホワイトペーパー

エグゼクティブ向けガイド：クラウドネイティブ開発プラットフォームの選び方

はじめに

組織でクラウドネイティブ・アプリケーションの開発計画を立てる際には、開発プラットフォームを選択する段階から、アプリケーションを本番環境レベルに仕上げてクラウドでの提供が可能な状態にするまでの開発期間全体を検討することが重要です。これは長期的なプロジェクトとなる可能性もあり、その過程に伴うさまざまな意思決定に応じて、スムーズに進行する場合も、進捗が妨げられる場合もあります。

たとえば、クラウドネイティブ開発への移行の初期段階において、アプリケーションのデプロイ先がまだ不明な段階で開発者がツールやフレームワークを選択し始めると、体制の非効率化につながりやすくなります。組織の開発者はランタイム、フレームワーク、言語に関して選択肢があることを望みますが、組織としては、運用コストの削減、リスクの緩和、コンプライアンス要件の遵守のために、アプリケーション・ライフサイクル全体を考慮した標準が必要となります。組織はさらに、クラウド・インフラストラクチャのプロバイダーについても、最新のアーキテクチャ・スタイルについても、ロックインを回避したいと考えます。

さらに、クラウド開発スキルの習得に必要な時間を踏まえ、堅牢性、スケーラビリティ、およびレジリエンシー (回復力) に優れたアプリケーションを構築するための考慮事項には、開発サイクルの早い段階で対応しておく必要があります。クラウド移行の決め手となる主要な要因は、動的スケーリングを可能とする弾力性であるため、特にレジリエンシーに関しては事後的にではなく、あらかじめ検討する必要があります。

効果的なクラウドネイティブ・アプリケーション戦略を実現するためには、開発ツールからクラウド・プラットフォーム、デプロイメントの自動化、運用に至るまでの全体像を考慮する必要があります。クラウドネイティブ開発に関する包括的なアプローチを採用することで、その手法を道標とし、不要な回り道を避ける助けとなります。

クラウドネイティブ開発について理解する

クラウドネイティブ開発の特徴は、スピード、レジリエンシー、俊敏性を活かして変化に対応することです。このような対応体制は、より頻繁にデプロイを行い、変化に対応するまでのリードタイムを大幅に短縮することで実現されます。

クラウドネイティブ・アプリケーションに関する議論の多くは、「Twelve-Factor App」というマニフェストに基づいています。これは、SaaS (Software-as-a-Service) アプリケーションの構築と運用の経験から集約された一連の原則であり、以下を目標としています。

- 開発者が学習に必要とする時間や労力の低減
- クラウド・プラットフォーム上へのデプロイに適したアプリケーションの構築
- 継続的デプロイによる俊敏性の最大化
- 大幅な変更を伴わずにアプリケーションのスケールアップを実現

Red Hat は、お客様をサポートしてきた経験から、クラウドネイティブ・アプリケーション開発における以下の重要な要素を見極めました。

- **サービスベースのアーキテクチャ**：このスタイルは、マイクロサービスでも、疎結合されたモジュール式のアーキテクチャモデルでもかまいません。サービスとは、容易に理解でき合理的に自己完結できるビジネス・アクティビティのことです。目標は、サービスを簡単に更新または交換できるようにすることです。サービスを適切に定義すると、完全なアプリケーションよりも、綿密なテストを実施しやすくなります。
- **コンテナ**：Docker イメージを使用する Linux® コンテナは、一般的なパッケージ化モデルであり、可搬性と分離性を備えた自己完結型の実行環境です。コンテナは高度な自動化を可能にし、クラウド・プラットフォームの魅力を増します。
- **DevOps 自動化**：開発チームと運用チームの統合を目指す、コラボレーションのプロセスとプラクティスです。デプロイ頻度の向上と提供するリリースの品質向上を目標としており、市場投入時間の短縮、リスクの軽減、およびユーザー満足度の向上につながります。DevOps が実現する改善は、継続的インテグレーションおよび継続的デリバリー (CI/CD) と密接に結びついています。また、計測と監視を実施することでパフォーマンスを把握し、優れたエンドユーザー・エクスペリエンスを確保することも、重要な目標の 1 つです。
- **API ベースの通信**：プロセス間通信は、ネットワーク上の契約ベースのクリーンなインタフェースを使用するアプリケーション・プログラミング・インタフェース (API) を介してのみ行われます。これにより、ダウンタイムの原因として一般的な、変更を制限する意図しない結合を防ぐことができます。これは、アプリケーションが同じデータセンター内に配置されないハイブリッド環境において、より重要となります。

マイクロサービスについて

マイクロサービス・アーキテクチャは、ソフトウェア開発の多くの課題を解決する魅力的なアーキテクチャ・スタイルです。アプリケーションは、特定のビジネス機能を実装する、一連の疎結合のサービス群へと分割されます。主要な目標はイノベーションの迅速化であり、これは「個々のマイクロサービスは、モノリシック・アプリケーションよりはるかに簡単に理解、改善、テスト、デプロイできる」という考え方に基づいています。各マイクロサービスの目的と機能が明確に定義されるため、自動化されたテストと継続的デリバリーの実装がはるかに現実的となります。

クラウドネイティブ・アプリケーションの要件

クラウドネイティブ・アーキテクチャは、アプリケーションにいくつかの技術的要件を課します。要件には以下のようなものがあります。

- **デプロイの自動化**：リリースの頻度を上げるには、デプロイの自動化を組み込み、承認ボタンをクリックするだけでプロセスが実行されるようにする必要があります。必要に応じて過去のリリースに迅速にロールバックできる機能とともに、自動化されたビルドとデプロイのパイプラインが必要です。
- **構成情報とアプリケーションの分離**：特定の環境内でアプリケーションを実行するために必要な構成の詳細は、デプロイ環境に保管しておく必要があります。これにより、開発、テスト、本番を含むすべての環境で同じアプリケーション・イメージを使用できます。さらに、アプリケーションはネットワーク上でデータソースやその他のコンポーネントと安全に通信する必要があるため、認証情報用のセキュリティ保護されたストレージも必要となります。
- **アプリケーション・サービスの動的な配置**：環境間の可搬性や、高可用性、負荷分散を使用する動的スケーリングを実現するために必要です。
- **永続データ用の別個のデータストア**：プロセスとコンテナはステートレスでなければなりません。プロセスはクラッシュするおそれがあり、数秒以内に (場合によっては別のマシンで) 再開される場合があります。データ、セッション情報、およびログは、すべて外部データストアに保存しておく必要があります。

クラウドネイティブ開発を開始する際の考慮事項は以下のとおりです。

- 組織では、クラウド・プラットフォーム上で動作するコンテナ化アプリケーションの構築とデプロイの経験がどの程度ありますか？開発者は、永続的なデータと構成情報を、アプリケーションとは別に管理するために何が必要であるかを理解していますか？
- クラウドネイティブ開発を開始するための開発環境の構築には、どの程度の時間がかかりますか？開発環境は本番環境のクラウドに対応していますか？
- CI/CD 体制を確立し、開発とデプロイのプロセスに効果的に組み込むまでには、どの程度の時間がかかりますか？

クラウドネイティブ開発プラットフォームを選択する際には、これらすべてを考慮する必要があります。

マイクロサービスに関する考慮事項

マイクロサービスとクラウドネイティブ・アプリケーションは密接に関連しています。実際、クラウド・プラットフォーム、コンテナ、および DevOps 自動化なしにマイクロサービスを実装することは困難です。マイクロサービスへの移行の主要な動機は、より小規模のアプリケーション・コンポーネントを使用することによって得られる俊敏性です。これにより、より頻繁かつ低リスクでリリースを行えるようになります。一方で、かつては単一のアプリケーションだったものが、それぞれ独立したリリースサイクルを持つ十数個ものマイクロサービスとなる場合があります。以下は、マイクロサービスに移行する際の考慮事項です。

- 新しいマイクロサービスごとに新しい開発環境、テスト環境、本番環境をプロビジョニングするにはどの程度の時間がかかりますか？
- ビルドとデプロイの自動化は、必要なリリース数をサポートするのに十分ですか？
- アプリケーションはマイクロサービスの実行場所をどのように検出しますか？可用性やスケーラビリティを確保するため、必要に応じてマイクロサービスを再配置できる柔軟性はありますか？
- エンドユーザーに影響を及ぼすダウンタイムを生じさせることなく、マイクロサービスをリリースできますか？
- 問題を診断することの難易度はどの程度ですか？さまざまなサービスやハイブリッド環境などでイベントをトレースすることの難易度はどの程度ですか？
- 1 つのサービスに関する問題は、十分に隔離されますか？それとも、複数の障害につながりますか？
- マイクロサービスは性質として分散しているため、サービス検出、クライアントサイドの負荷分散、永続的な状態管理、分散トレーシング、耐障害性などの機能を実行するためには、追加のソフトウェア・インフラストラクチャ・コンポーネントが必要です。
- 開発者はこれらのサービスを実装する必要がありますか？それともインターネットでコンポーネントを見つけられますか？
- これらのコンポーネントが、現在デプロイ済みのマイクロサービスすべての一部となるとして、保守の負担はどの程度になりますか？バグやセキュリティ上の脆弱性に対処するにあたって、コンポーネントはどのように更新されますか？
- これらはクラウドランタイムまたはプラットフォームに組み込む必要のあるコモディティサービスですか？

開発者がマイクロサービスを構築するにあたって、最初に生じてくる疑問の 1 つは、どのような種類のランタイム、フレームワーク、言語が必要であるかということです。この疑問を解消するための考慮事項は以下のとおりです。

- 開発者は新しいフレームワークについて学習する必要がありますか?
- マイクロサービスに移行する際、組織内の既存の Java™ EE コードと専門知識を活用することはできますか?
- リアクティブ・プログラミングなどの新しい開発スタイルを使用することはできますか?プラットフォームには、モバイルや IoT (モノのインターネット) で作成されたイベント駆動型ワークロードなど、新たな要求を満たすために必要な選択肢が用意されていますか?
- 作業内容に最適なツールを使用するために、さまざまなランタイム、フレームワーク、または言語を使用してさまざまなマイクロサービスを実装することが合理的でしょうか?設定、デプロイ、セキュリティ保護の方法に相違点はありますか?
- 実際のパブリッククラウド環境で新しいマイクロサービスをテストするには、どの程度の時間がかかりますか?

マイクロサービスは非常に魅力的です。しかし、マイクロサービスと分散アプリケーション・アーキテクチャには、習得が難しいという側面があります。そのため、複雑さを排除し、開発作業を単純化できるプラットフォームを選択することが重要となります。

既存のモノリシック・アプリケーションのモダナイゼーション

既存のモノリシック・アプリケーションをマイクロサービスに書き直すことは、開発者にとっては魅力的かもしれませんが、実用性やコスト効率に欠く場合もあります。しかし、既存アプリケーションにはいずれにせよメンテナンスが必要であり、実装が必要な改良のバックログも存在します。リリースサイクルが長い場合、バックログの削減は困難になりがちです。これらのアプリケーションをクラウド・プラットフォームにリホストすれば、CI/CD の実践に加えて、ローリングリリース (ブルーグリーン・デプロイメント、カナリア・デプロイメントなど) の実現も容易になります。これにより、開発者はより低リスク・高頻度でリリースを提供できるようになります。クラウドでホストされているアプリケーションが「高速で動作するモノリス」となり、機能改良のバックログへの対応がより容易になります。

アプリケーションをクラウドにリホストすることは、マイクロサービスの実装に向けた効果的な第一歩にもなり得ます。クラウドは柔軟性に優れているため、新しいマイクロサービスを既存アプリケーションとともにコンテナにデプロイすることが、より容易になります。その後、モノリシック・アプリケーションから機能を移行するマイクロサービスを実装できます。

既存アプリケーションをクラウドに「リフト & シフト」方式で移行するメリットを踏まえると、クラウドネイティブ開発プラットフォームを選択する際の考慮事項は以下のようになります。

- その開発プラットフォームには、完全な新規開発を行う用途だけでなく、既存アプリケーションをモダナイズするためのオプションも用意されていますか?
- そのクラウド・プラットフォームは、まだ完全にクラウドネイティブではないアプリケーションをサポートできますか?
- 既存の Java EE アプリケーションの移行方法にはどのような選択肢がありますか?
- その開発プラットフォームはモノリス式アプリケーションのデリバリーを加速しますか?

アプリケーション・サーバーは、クラウドネイティブ・アプリケーションのランタイム環境として機能できますが、従来のアプリケーション・サーバーの役割はクラウドネイティブ・プラットフォームに移行する際に変わります。従来のアプリケーション・サーバーは、ランタイムとデプロイメント環境、およびシステムクラスタで実行される中央ドメインの管理サービスを提供していましたが、含まれている運用ツール (管理コンソールなど) の多くは不要となり、むしろ生産性を阻害するようになります。クラウド・プラットフォームが魅力的である理由の 1 つは、動的スケーリングと継続的デリバリーを可能にする自動管理機能が含まれていることです。従来のアプリケーション・サーバーを管理コンポーネントとともにコンテナにパッケージ化することで、開発者はこれらの機能を利用できなくなります。以下の事項を考慮する必要があります。

- そのプラットフォームには、クラウド・プラットフォームの管理機能と統合されている Java EE ランタイムのオプションが含まれていますか?
- 従来のアプリケーション・サーバー上で実行されているアプリケーションを最新のクラウドベースの Java EE 環境に移行するためのツールは用意されていますか?
- Java EE のすべての機能を使用しないアプリケーション (Web アプリケーションなど) の場合、よりクラウドに適した代替手段はありますか?

適切なプラットフォームとツールを使用することで、クラウドネイティブとマイクロサービス・モデルに移行しながら、既存アプリケーションの価値をさらに引き出すことができます。

オンプレミスとマルチクラウドのサポート

大半の IT 組織は、1 つだけのパブリッククラウド・インフラストラクチャ・プロバイダーに拘束されないことを望んでいます。また、多くの組織が、数年後においても、アプリケーションの半数以上は引き続きオンサイトで実行されるものと考えています。ほとんどの組織にはこのような 2 つの要件があるため、以下を考慮する必要があります。

- 異なるプロバイダーのクラウド・インフラストラクチャを使用するために、アプリケーションの変更が必要になりますか?デプロイメント、運用、監視についてはどうですか?それらはクラウド・プラットフォームごとに異なりますか?
- パブリッククラウド・インフラストラクチャとオンサイトシステムの相違を最小限に抑える方法はありますか?パブリッククラウドで使用されるクラウド・プラットフォームは、オンサイトでも、組織が使用したいと考える IaaS (Infrastructure-as-a-Service) でも利用できますか?

視野の拡大

クラウドネイティブ開発プラットフォームを評価する際は、その選択が他の分野にどのように影響するかを考慮することにも価値があります。

- クラウドの開発およびデプロイメント・プラットフォームすぐに使用できる開発ツールは提供されていますか?すぐに使用できるツールがない場合、開発者がツールを設定し統合するのにどの程度の時間がかかりますか?
- そのプラットフォームは、生産性を向上させるための規範や道標となるアプローチを開発者に提供しますか?
- メッセージング、データストレージ、ビジネスプロセス、またはルール管理用のクラウドベースのミドルウェアサービスが利用可能であり、クラウド・プラットフォーム上で実行できる状態になっていますか?
- 事前構築されたサードパーティ製のコンテナをアプリケーションと統合することはできますか?
- トレーニング・サービスやコンサルティング・サービスは利用できますか?配置可能な社内の人員に、アプリケーションのモダナイゼーションの経験はありますか?

適切なクラウド開発プラットフォームを選択する重要性

クラウド開発を開始する開発者は通常、チュートリアルに従って、インターネットからソフトウェア・コンポーネントを選択することによって、一度に 1 つずつ問題を解決していきます。まず最初に、アプリケーション・コードをコンテナとしてビルドできるランタイム環境をアセンブルする方法を習得する必要があります。次に、完全なアプリケーションを構築するために必要な他のソフトウェア・コンポーネントを決定する必要があります。このプロセスは時間がかかる可能性があり、他にも以下のようなデメリットがあります。

- 習得にかかる時間およびどの程度の統合作業が必要かに応じて、開発者の生産性を損なうおそれがあり、このことを企業に説明するのは困難です。新たに採用した開発者は、社内でこれまで確立されてきた大量の手法を習得する必要があります。
- 選択したソフトウェア・コンポーネントは、既知のバグやセキュリティ上の脆弱性がない状態が維持されるように保守する必要があります。オープンソースのコンポーネントには、企業での使用に適しているかどうかという問題もあります。
- 多くの組織が、開発者は、ビジネス価値を直接もたらさない構成やデプロイなどの機能に関するコードの記述を継続しない場合があると報告しています。さらに困ったことに、これらの機能の多くはデプロイメント・プラットフォームの選択と密接に関係しており、デプロイメント・プラットフォームの一部であるべきコモディティサービスでもあります。

多くの場合、コンポーネントのコレクションをアセンブルする上での最大の問題は、それが開発と運用の全体的な学習難度の緩和に貢献しないことです。そのため、代替手段として、ランタイムの選択から開発の開始に至るまで、本番環境へのデプロイメントまでのあらゆる段階を処理するように設計されたクラウド開発プラットフォームを選択する必要があります。

クラウドネイティブ・アプリケーションへの移行は、一晩で完了するプロセスではありません。多くの組織にとってこれは、経験を積み重ねることで改善されていくプロセスです。クラウドネイティブ開発に伴うあらゆる要素を考慮すると、開発からデプロイメントまでをカバーするエンドツーエンドのアプローチが成功する可能性が高いことは明らかです。Red Hat は、Red Hat® OpenShift Application Runtimes と Red Hat OpenShift を備えたプラットフォームを提供しています。

RED HAT OPENSHIFT APPLICATION RUNTIMES と RED HAT OPENSHIFT

Red Hat OpenShift Application Runtimes は、クラウドネイティブ・アプリケーションの開発を単純化するように設計されています。厳選された一連の統合済みランタイムとフレームワークが、道標となるアプローチと規範的なエクスペリエンスを提供し、開発プロジェクトをスムーズに始動させるための助けとなります。OpenShift Application Runtimes は、ハイブリッドクラウド用に設計されたコンテナ・アプリケーション・プラットフォームである Red Hat OpenShift をベースとしており、それに合わせて最適化されているため、開発とデプロイメントのプラットフォームが完全に効率化されます。

Red Hat OpenShift は、コンテナ化されたアプリケーションを構築して実行するための開発者向けセルフサービス・プラットフォームを提供します。OpenShift を使用すると、新しいマイクロサービスまたはアプリケーションの環境をわずか数分でプロビジョニングできます。OpenShift なら、デプロイメントサイクルを大幅に短縮し、リスクを大幅に削減することが可能です。わずか数回クリックするだけで、自動化された強力な CI/CD のビルドおよびデプロイメント・パイプラインが実現します。Red Hat OpenShift Application Runtimes と Red Hat OpenShift、Red Hat 開発ツールと Red Hat コンサルティングを組み合わせて使用することによって、時間とリスクを最小限に抑えてクラウドネイティブ・アプリケーションに移行することができます。

RED HAT® OPENSHIFT

redhat.

OpenShift Application Runtimes のランタイムを選択すると、作業に適したツールを開発者に提供することができます。マイクロサービス開発の場合、開発者には、Java EE を使用して既存の専門知識を活用するか、マイクロサービスのニーズを満たすように進化した新しい Java マイクロプロファイル標準を使用するか、あるいは高い同時実行性、低レイテンシのワークロードに対応するリアクティブ・マイクロサービスを構築するイベント駆動型フレームワークを使用するか、という選択肢が与えられます。モバイルや Web アプリケーションとともに頻繁に使用される JavaScript バックエンドサービス用には、Node.js ランタイムが用意されています。既存アプリケーションの移行のためには、Red Hat JBoss® Enterprise Application Platform をベースとしたランタイムが用意されています。これは、先進的なモジュール式のクラウド対応アーキテクチャに基づいた Java EE 7 認定アプリケーション・サーバーです。これらすべてのランタイムは Red Hat によりテストおよび検証済みです。

Red Hat OpenShift Online で動作する無料のクラウドベースの SaaS ツールである Launch は、Red Hat Developers の Web サイトから入手でき、開発者が迅速に作業を開始できるように支援します。開発者は、サンプル・アプリケーションとランタイムを選択することで、クラウドの OpenShift Online 上で構築および実行できる、完全なコードベースを利用できます。教育ツールとして、すべてのランタイムで同じサンプル・アプリケーションを使用できるため、開発者は使用可能なアーキテクチャ・スタイルのメリットを簡単に比較できます。

詳しくは、https://www.redhat.com/ja/technologies/cloud-computing/openshift/application-runtimes をご覧ください。

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

オーストラリア
1 800 733 428

ブルネイ / カンボジア
800 862 6691

インド
+91 22 3987 8888

インドネシア
001 803 440224

日本
03 5798 8510

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

フィリピン
800 1441 0229

シンガポール
800 448 1430

タイ
001 800 441 6039

ベトナム
800 862 6691

中国
800 810 2100

香港
852 3002 1362

台湾
0800 666 052

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

jp.redhat.com
F10657_V1_0118_KVM