ホワイトペーパー

成功するマイクロサービスアーキテクチャの実現

利点

- 開発および管理の簡素化
- 迅速で独立したデプロイ
- アプリケーションコンポーネントの独立した拡張性
- テクノロジーの長期契約からの解放

要旨

マイクロサービスアーキテクチャは、疎結合でありながら自立したサービス向けの新しいスタイルのアーキテクチャです。DevOps、PaaS (Platform-as-a-Service)、コンテナや、継続的な統合および提供 (CI/CD) 手法などのテクノロジーにおける新たなトレンドでは、組織はサービス指向アーキテクチャ (SOA) のような初期のアプローチを越える前例のない規模で、モジュラーシステムを作成、管理することができます。しかし、単一のアプリケーションをマイクロサービスにリファクタリングする組織が経験する成功の度合いは、実にさまざまです。マイクロサービスを効果的に利用するのに重要なのは、組織がマイクロサービスを使ってアプリケーションを構築する方法とその理由を確実に理解することです。

サービス指向アーキテクチャの改善

サービス指向アーキテクチャ (SOA) は、ネットワーク経由で他のコンポーネントにサービスを提供するために通信するアプリケーションコンポーネントとして一般的に定義されます。SOA の目標は、複雑な集中型のコンポーネントのない回復力の高い分散型アプリケーションを作成することでした。

ところが、SOA は組織構造に強く結合され、新しい内部構造に対応するために応用されました。そのため、SOA の成功は、結果として生じる再構築された組織能力およびアーキテクチャを設計するチームの構造に大きく依存していました。疎結合でありながら自立したシステムを作成する代わりに、SOA では複雑なインフラストラクチャを必要とするきわめて脆弱なシステムが作成されました。さらに、初期の SOA 実装ではプロプライエタリなミドルウェアが、集中型のロジック、永続化、ガバナンス、管理を重視することが多いため、ベンダーロックインの問題が生じました。

マイクロサービスアーキテクチャは、開発から展開、運用にいたるまで、アプリケーション構築のあらゆる段階で SOA が約束されていることを実現し始めています。マイクロサービスアーキテクチャでは、合理化されたコンポーネントを利用して複雑なシステムを構築するテクノロジーを簡素化することに重点が置かれています。標準化されていない大規模なプラットフォームを基盤にした集中型ロジックおよび統合インフラストラクチャは、非同期 HTTP またはメッセージングプロトコルを基盤とする標準化されたシンプルなパイプを介したコミュニケーションに置き換えられています。SOAP、XML、およびその他の非軽量プロトコルやデータ形式は、HTTP ベース REST を介した軽量の JSON に置き換えられています。各マイクロサービスには固有のデータストアがあり、集中型のガバナンスおよび永続化は必要ありません。

マイクロサービスでは、継続的な統合 (CI) および継続的な提供 (CD) の方法論と実践に加え、SOA では一般的でない次のような重要なコンポーネントも使用されています。

- 多言語対応プログラミングおよび永続化
- コンテナまたは不変仮想マシン (VM)
- 柔軟性のあるプログラム可能な IaaS (Infrastructure-as-a-Service) および PaaS

マイクロサービスに対する準備はできていますか?

組織では、

- 構造化に優れた単一のアプリケーションが構築されているか?
- マイクロサービスによって満たされるニーズが特定されているか?
- マイクロサービス関連のチームが再編されているか?
- DevOps および CI/CD にベストプラクティスを採用しているか?
- アプリケーション内でビジネスの境界が特定されているか?
- マイクロサービスオーケストレーションおよび管理ツールとプロセスが実装されているか?

IT の柔軟性と応答性を向上させる革新的ソリューション

デプロイの高速化

マイクロサービスはドメイン境界および一貫したドメインモデリングを重視することによって範囲が特定されるため、その範囲は小さく、必要なコードも少なくてすみます。集中的な自己完結型アーカイブを含むデプロイ戦略は、Linux コンテナおよび CI/CD としてパッケージ化されることが多いため、より信頼性の高い、高速デプロイにつながります。その結果、一般的にソフトウェア開発のライフサイクルは短縮されます。新機能およびバグ修正に加え、十分にテストされたセキュリティパッチもより迅速にリリースされます。

モジュラー制御

マイクロサービスにより、各サービスは一時的または季節的な要因によるトラフィックの増加、一括処理の実行、およびその他のビジネスニーズに対応するために個別に拡張できます。障害分離の改善により、メモリリークやオープンデータベースの接続などのサービス上の問題が特定のサービスのみに影響するよう制限されます。マイクロサービスの拡張性によってクラウドサービスの柔軟性が補完されるため、サービスが向上するだけでなく、サービスを中断せずにより多くの顧客に対応できます。

多様な選択肢

オープンソーステクノロジーソリューションおよび組織的手法は、マイクロサービス市場をリードしています。その結果、マイクロサービスによってベンダーロックインが軽減され、テクノロジーの長期契約が不要になるため、IT およびビジネスの目標の達成に必要なツールを選ぶことができます。

マイクロサービスの堅牢な基盤の構築

マイクロサービスで成功を実現するために、組織ではまず単一のアーキテクチャの堅牢な基盤を構築する必要があります。マイクロサービスの利点をフルに活用するためには、モジュール化、ドメイン境界、基本的な分散システムの理論を考慮して確立する必要があります。

さらに、マイクロサービスによって、より複雑なシステムにおいても最大限の利点が生まれます。各サービスは完全に独立していますが、次のような機能を含めた、運用要件を満たす必要もあります。

- DevOps
- PaaS
- コンテナまたは不変 VM
- サービスのレプリケーション、レジストリ、および検出
- プロアクティブな監視および警告

このような要件を満たすことは、多額の投資を行ってもすぐに回収できない場合があるため、マイクロサービスはすべてのチームまたはプロジェクトにとってコスト効率に優れているわけではありません。モノリスファーストと言われるアプローチを評価することで、堅牢な設計原則に従ってアプリケーションが構築され、ドメイン境界が適切に定義されるようになるため、拡張が必要な場合は、徐々にマイクロサービスアーキテクチャに移行することができます。たとえば、基本的なオンラインショッピングアプリケーションには、次のものが含まれている必要があります。

- 問題の分離
- 明確に定義されたアプリケーションプログラミングインターフェース (API) による高結束および疎結合
- デメテルの法則 (最小知識の原則) によるインターフェース、API、および実装の分離
- 関連するオブジェクトをグループ化したドメイン駆動型の設計

組織がマイクロサービスによって成功する理由は、その組織構造によるもので、組織のテクノロジーによるものではありません。フラットな組織構造で柔軟に対応できるチーム、部門間の協力体制、そして自主性が成功のカギです。

堅牢なソフトウェアアーキテクチャ原則に沿って、拡張が必要な単一のアプリケーションが構築されると、マイクロサービスにリファクタリングできます。最も効果的なリファクタリング手法は、次のステップで構成されます。

1. ビジネスの境界とアプリケーションドメインにおける定義の違いを特定し、各ドメインを固有のマイクロサービスに分解する
2. 価格計算や規制の変更に伴うビジネスルールの更新など、要求の最も多い変更によって影響を受けるコンポーネントや、セキュリティの脆弱性を解決するためのパッチが頻繁に適用されるコンポーネントを見つける
3. 基本的なドメインベースのマイクロサービスを定義した後で、サービスのやり取りに使用される API を改善する。アグリゲーター、プロキシ、チェーン、分岐、イベント駆動型の設計やその他の設計パターンを使って API を構成する

高いスキルを持つチーム間の連携

組織がマイクロサービスによって成功する理由は、その組織構造によるもので、組織のテクノロジーによるものではありません。フラットな組織構造で柔軟に対応できるチーム、部門間の協力体制、そして自主性が成功のカギです。

効果的で高いスキルを持つチームを構成するには、アーキテクチャではなく機能を中心としたスタッフの再編成が必要です。たとえば、Amazon が提唱する 8 人から 10 人のスタッフでサービスの開発と維持を担当する「2 枚のピザチーム」は、その代表的な例です。コンウェイの法則では、組織は組織の構造を模倣した設計しか生み出すことができないと結論付けられています。たとえば、開発、運用、品質保証、セキュリティなどに分割されたチームでは、柔軟性が限定され、進捗に遅れが生じます。

マイクロサービスに移行する前に DevOps 手法を確立することで、このような問題が軽減され、未然に防ぐことができます。また、事前にコミュニケーション戦略を特定することで、SOA の障害が回避され、効果的なマイクロサービスアーキテクチャが生まれます。

効果的な管理ツール

よく設計されたソフトウェアおよび効果的に編成されたチームに加え、拡張性に優れたアーキテクチャの実現には、追加のサービスやアプリケーションコンポーネントを管理するための次のようなツールが必要です。

- サービスレジストリおよび検出ツール (Kubernetes など)
- アプリケーションをコンテナ化するための Docker などのパッケージ化規格、およびコンテナを規模に応じてレプリケートするための Kubernetes などのオーケストレーションツール。OpenShift by Red Hat は、これらの実証済みオープンソーステクノロジーを含む
- Docker および Kubernetes 向けの Jenkins や Shippable などの CI 環境構築ツール
- Nexus などの依存性解決ツール
- Hystrix や Ribbon などのライブラリを含むフェイルオーバーおよび耐障害性ツール
- ELK (ElasticSearch、LogStash、および Kibana) スタックなどのサービスモニタリング、警告、およびイベントツール

データ管理

マイクロサービスへの移行に関して考慮すべきもう一つの重要な事項は、データ管理です。SOA とは異なり、マイクロサービスはデータを共有しません。その代わり、各マイクロサービスには個別の物理データストアがあり、多言語対応の永続化によって、各マイクロサービスの下でさまざまなデータベースエンジンを実行できます。その結果、すべてのデータを企業のリレーショナルデータベース管理システム (RDBMS) に保存する代わりに、サービスごとにデータストアを選ぶことができます。

ただし、エンタープライズデータベースの複数のコピーを管理することになるため、ライセンスコストが増加し、複雑さが増大する場合があります。さらに、一貫性を保つためにデータストアの調整が必要になる場合があります。汎用の抽出/変換/ロード (ETL) ツールやデータ仮想化ツールはデータの正規化に効果があり、イベントソーシングはデータストアを遡及的変更に合わせて調整できる、よく知られた設計パターンです。

まとめ

マイクロサービスアーキテクチャの導入によって、多様なアプリケーションコンポーネントの個別の拡張性から、迅速で容易なソフトウェア開発と維持にいたるまで、多くの利点が組織にもたらされる可能性があります。しかしながら、マイクロサービスはあらゆるチームやプロジェクトにとって必ずしも利点があるというわけではなく、多額の投資を行ってもすぐに回収できない場合があります。マイクロサービスへの移行は段階的に進める必要があり、完全移行ではなく、既存のアプリケーション部分のみをリファクタリングすることでも利点が生まれます。マイクロサービスによる成功を実現するために、組織は既存のプラットフォーム基準に沿って、まずよく設計されたアプリケーションを構築してから、必要に応じてビジネスニーズに合うように、マイクロサービスの集合体にアプリケーションをリファクタリングする必要があります。適切な人材、プロセス、およびツールを活用することで、マイクロサービスによる迅速な開発と展開、管理の簡素化、拡張性の向上、テクノロジーの長期契約からの解放を実現できます。

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
INC0336100_0216