マイクロサービスとは？ そのメリットを簡単に解説（初心者・非エンジニア向け）

公開 : 2020.03.10　最終更新 : 2020.03.17

カテゴリー　システム開発　　タグ　AWS　クラウド

NCDCのマーケティング担当、播磨です。

先日、社内のエンジニア勉強会で「マイクロサービス」が取り上げられていました。
僕自身はエンジニアではないので、少し前まで「マイクロサービスって何？　それ食えんのか？」というレベルの知識しか持っていませんでしたが、勉強会の資料を参考にして少し学んでみたので、非エンジニアでもわかるコンテンツとしてアレンジしたものを公開します。

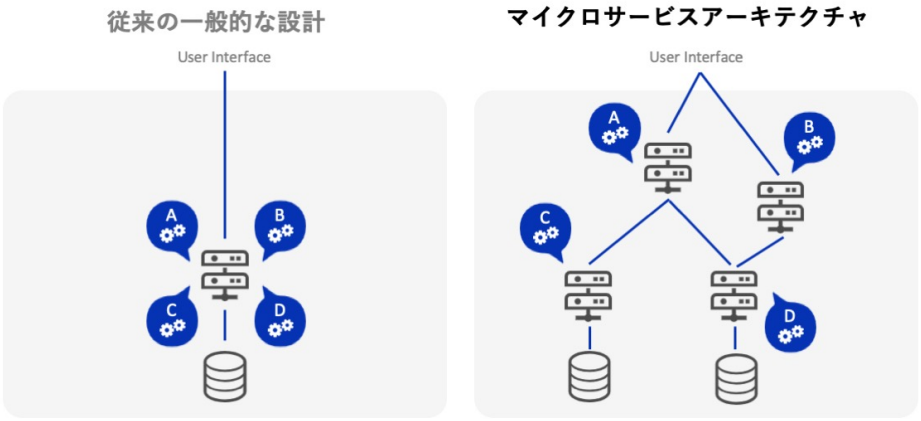
（具体的にマイクロサービスの導入を検討されている方はマイクロサービス導入支援のサービス紹介ページもぜひご覧ください）

目次　[非表示]

マイクロサービス（マイクロサービス アーキテクチャ）とは？

マイクロサービスとは、複数の規模の小さなサービスを組み合わせてひとつの大きなアプリケーションを構成する、ソフトウェア開発の技法のひとつです。
従来は、「すべての機能がまとまったひとつの大きな塊」としてソフトウェアを設計することが多かったのに対し、マイクロサービスは「まず機能を分解していき小さなサービスをつくる。それらを組み合わることでひとつの大きなソフトウェアを構成する。」という考え方です。

ひとつひとつのサービスは小さいだけでなく、自らの持つ役割に専念して、自律的に機能するという点もマイクロサービスに欠かせない要素です。
具体的には、個々のサービスはできるだけシンプルにすること、そして複数のサービスがひとつのOSやハードウェアの上で動かないようにすることが重要です。
その理由は後述しますが、とにかく個々のサービスは依存関係が低く、それぞれのサービスの呼び出しはネットワークを介して行われるというのもマイクロサービスの特徴です。



内部で機能A,B,C,Dに分かれていても、基本的にひとつのマシン上で動く単一の構成。

機能A,B,C,Dは独立したマシン上で動く。
それぞれはネットワークを介して呼び合う。

マイクロサービスは、置き換えるとアメーバ経営みたいなもの？

「小さく分けて、それぞれが自律的に動くようにする。」どこかで聞いたことがあるなと思ったら、考え方としてはアメーバ経営と同じですね。

> アメーバ経営では、組織をアメーバと呼ぶ小集団に分けます。各アメーバのリーダーは、それぞれが中心となって自らのアメーバの計画を立て、メンバー全員が知恵を絞り、努力することで、アメーバの目標を達成していきます。そうすることで、現場の社員ひとりひとりが主役となり、自主的に経営に参加する「全員参加経営」を実現しています。
>
> 出展：アメーバ経営 | 稲盛和夫 OFFICIAL SITE

「マイクロサービス」の対義語は「モノリシック」

「マイクロサービス」の対義語としては「モノリシック」という言葉があります。
モノリスは「一枚岩」という意味の英語で、ソフトウェアの世界では大きな岩の塊のように全体がひとつのモジュールになっている構造をモノリシック アーキテクチャと表現するそうです。

余談ですが、モノリスといえば映画『2001年宇宙の旅』（監督：スタンリー・キューブリック,1968）に登場する、あの石柱状の謎の物体を思い出します。

2001年宇宙の旅のモノリス（真ん中の黒いやつ！）

『2001年宇宙の旅』は子どもの頃に見たので、「モノリス」という名称を何も考えず受け入れていましたが、「モノリス」とは見た目そのままに一枚の岩（板）という意味だったのですね。
自分が生まれる前につくられた映画なので何年前に見たか定かではないですが、「モノリス」という言葉を見たらすぐ思い出せるくらい「モノリス＝謎の石版」という印象が強く残っています。
余談おわり。

どこまでサービスを小分けにすればマイクロサービスなのか？

本題に戻ります。
マイクロサービスとは「小さなサービスを組み合わせる」ソフトウェアの作り方だと書きましたが、具体的に各サービスをどのくらい小さくするとマイクロサービスといえるのか？
マイクロサービスとは何かという正確な定義はないようなのですが、書籍『マイクロサービスアーキテクチャ』（Sam Newman (著), 佐藤 直生 (監修), 木下 哲也 (翻訳)）にはこのような記述があります。

> サービスが小さければ小さいほど、マイクロサービスアーキテクチャの利点と欠点が最大化されます。小さくすればするほど、相互依存関係に関わる利点が増加します。しかし、可動部が増えることで生じる複雑さも増します。
> （中略）
> この複雑さへの対応がうまくなると、さらに小さなサービスを追求することができます。

要は、開発思想としてマイクロサービスを取り入れる場合、できるだけ細かくサービスを分割する方が望ましい（マイクロサービス アーキテクチャの目指すべき姿といえる）。その一方で、構造が複雑になってしまうというデメリットも起こり得るため、そのバランスを考慮して取り組む必要があるということですね。

結局、どこまで細かくサービスを分割するべきかについては、一概に「これが正解」といえるものはないため、「細かくサービスを分割し、それぞれが自律する」という開発思想で取り組めば、それはマイクロサービス アーキテクチャだといえるようです。

マイクロサービスのメリットを簡単に解説

「マイクロサービス」という言葉は2014年から提唱されはじめ、2019年の現在でもIT業界で注目のキーワードとなっています。
その理由は、やはり導入することで多くのメリットがあるからなのですが、具体的には何が得られるのでしょうか？

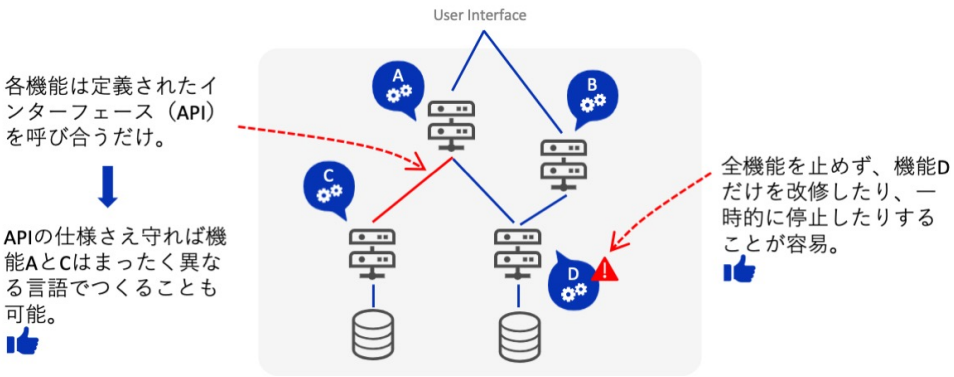
マイクロサービスの導入で何が得られるのか？

マイクロサービスはさまざまな新しい技術を容易に採用できることが大きなメリットといえます。
マイクロサービスでは、各々のサービスはネットワークを介してそれぞれが公開したインターフェースを呼び合うだけで、それ以外の制約、依存関係はありません。
そのため、相互に呼び合うためのインターフェース仕様を守っている限り、それぞれのサービスがまったく異なる技術を採用することもできるのです。

たとえば、新規システムをつくる際に、ある部分にはAという技術（開発言語）、ある部分にはBという技術（開発言語）が適しているとします。
モノリシックな構造の場合は、AかBかどちらか一方の技術を選択する必要がありましたが、マイクロサービスでは他の部分がどの技術を使っているかを気にせずにそれぞれのサービスにとって最適な技術を選択することができます（こっちはAの技術でつくる。こっちはBの技術でつくるという選択が自由に行えます）。

そうすることで開発期間の短縮が可能となるため、新機能の投入や新しい技術の採用が比較的容易になります。
それは新規開発のときだけではなく、システムの運用時に機能追加がしやすかったり、障害時の対応がやりやすかったりというメリットにも繋がります。

マイクロサービスアーキテクチャ



つまり、新しい技術の採用や、障害時の対応が比較的容易な柔軟性の高い仕組み。

もう少し大きな視点で見ると、システムの開発や機能追加が容易になるということは、そのシステムを利用するビジネスにおいても変化に対応しやすくなるというメリットにつながります。
モノリシック アーキテクチャよりもマイクロサービス アーキテクチャの方が柔軟性が高いと表現することもできます。
そう考えると、技術面の進化だけが理由ではなく、技術の進化に伴ってビジネス環境の変化も激しい時代だという背景があって、マイクロサービスという開発手法が注目されているのかもしれません。

マイクロサービスのさまざまなメリット

マイクロサービスのメリットもう少し細かくみていくと、前述したもの以外にもさまざまなポイントを挙げることができます。
AWSのWebサイトに解説があったので、それを引用しつつ、簡単に説明を加えたいと思います。
出展：マイクロサービスの概要 | AWS

> **俊敏性**
>
> マイクロサービスでは、サービスの所有権を持つ小規模で独立した複数のチームからなる組織が発展します。チームは、小規模でよく理解されたコンテキストにおいて行動し、より自主的かつより迅速に仕事に取り組む権限を与えられます。これにより、開発サイクルが短縮されます。組織の総スループットを大いに有効活用できます。

先ほども「開発期間の短縮」について触れましたが、その背景として「小さく、ひとつの機能に特化したサービス」は、開発もそれぞれの「小さな独立したチーム」で行えるということがあります。
各サービスに対して少人数の独立したチームが責任を持つことで、大規模な開発チームと比較して俊敏に対応できるようになります。

> **柔軟性のあるスケーリング**
>
> マイクロサービスを利用すると、サポートするアプリケーション機能の要求に応じて各サービスを個別にスケールできます。これによりチームは、インフラストラクチャに対するニーズの適正化、各機能にかかるコストの精確な評価、サービスに対する需要のスパイクが生じた場合の可用性の維持が可能です。

マイクロサービス アーキテクチャの場合、システム全体ではなく、必要な部分だけを対象としてスケーリング（処理能力の増強や縮減）を実行することができます。たとえば一部の機能で処理能力が限界に達したとしても、マイクロサービスの場合はその一部だけのために全体の処理能力を増強する必要がなくなるため、人、時間、金銭面のすべてにおいて、コストを抑えた対応が可能になります。

> **容易なデプロイ**
>
> マイクロサービスでは、継続的インテグレーションと継続的デリバリーが実現可能です。容易に新しいアイディアを試したり、上手くいかなかった場合にロールバックしたりできます。失敗時

のコストが抑えられるため、活発に実験を行えるほか、コードの更新が容易になり、新機能の市場投入を加速できます。

マイクロサービス アーキテクチャでは、個々のサービスを独立してデプロイすること（開発した機能やサービスを利用できる状態にすること）ができます。
また、問題が生じた場合は、問題のあるサービスだけを個別にロールバックする（障害発生時に正常に稼働していたある時点の状態に戻して復旧を試みること）などの対応が可能になります。
そのため、「１枚岩」でできているモノリシック アーキテクチャと比較して試験的な新技術の導入が行いやすく、障害への対応も比較的容易になるといえます。
また、個々のサービスを分けることで一部改修を行う際の影響範囲が少なくなるため、一部のサービスをより優れたコードに書き直したり、必要なくなったサービスを削除したりという保守作業を低コストで実行できるようになります。

> **技術的な自由**
>
> マイクロサービスアーキテクチャは、「フリーサイズ」のアプローチを踏襲しません。チームには、特定の問題を解決するための最適なツールを選ぶ自由があります。その結果、マイクロサービスを構築するチームは、各ジョブに最適のツールを選択可能です。

先にも触れた通り、マイクロサービスは基本的に他のサービスがどのような技術でつくられているかによって制約を受けたり、依存関係を持ったりしないため、どんな技術を採用するかは、そのサービスを担当するチームの自由ということになります。
一方で、モノリシックなソフトウェアの場合は大規模なチームで技術を統一する必要があるため、どうしても「この機能だけを見れば本当は他の技術でつくる方が楽だけど、他の機能との関係上我慢するしかない」というが制約が出てきてしまうのだといえます。

> **再利用可能なコード**
>
> ソフトウェアを小さな正確に定義されたモジュールに分割すると、チームは複数の目的に合わせた機能を使用できます。ある機能のために記述されたサービスは、別の機能のために構成要素として使用できます。そのため、アプリケーションはそれ自体から独立でき、開発者は一からコードを記述することなく新機能を作成できます。

機能が細分化されているため、必要な機能だけを取り出して別のサービスで再利用しやすくなります。
たとえば「注文を受けて、在庫から探し出し、出荷する」という一連の機能を持つシステムをつくる場合に
・注文管理
・在庫管理
・出荷管理
と3つのサービスに分けておくことで、「在庫管理のサービスだけ他のアプリケーションに再利用しよう」ということが容易にできるようになります。
（こちらはあくまでも説明用の例です。実際の設計としては「注文管理」の中でもさらに細かいサービスに分解していくのだと思います）

> **耐障害性**
>
> サービスの独立性により、アプリケーションの耐障害性が向上します。モノリシックアーキテクチャでは、１つのコンポーネントに障害が発生すると、アプリケーション全体に障害が及ぶおそれがあります。マイクロサービスでは、アプリケーションの機能性を低下させてアプリケーション全体のクラッシュを回避することにより、全体的なサービス障害に対応します。

モノリシックなソフトウェアは一つの障害で全ての機能が止まってしまいますが、マイクロサービス・アーキテクチャを採用している場合、その障害が連鎖しなければ部分的に機能し続けることができます。

こうして見るといいことづくめですね。もはや「マイクロサービス最高！」としか言いようがないように思えます。
ただし、もちろんデメリットがまったくないわけではありません。

マイクロサービス にデメリットはないのか？

前述したように、マイクロサービス アーキテクチャでは小さいサービスをたくさん連結させるために構成が複雑になるおそれがあります。
また「技術的に自由」で、サービスごとに異なる技術を使えるということは、裏を返せばそれを統括する立場の人はさまざまな技術の理解が必要になるという面もあります。

つまり、マイクロサービスのメリットを最大限に得るためには、幅広い知識と高度な技術力が必要といえます。
ただ流行っているからという理由だけで手を出すと、メリット以上のデメリットに直面してしまうおそれもあるので、マイクロサービスを適用する際は、この点についてもよく考えておく必要があります。

豊富な実績があるNCDCの技術コンサルティング

いかがでしょうか。非エンジニアの方にも「マイクロサービスってそういうことか」とわかってもらえる説明になっているといいのですが…。
（もともとはエンジニア勉強会の資料なので、もっと深く、具体的なモデルに言及した部分もあったのですが、深い話はかなり省略して基礎知識的な部分を手厚くしています。）

だいぶ長くなりましたので、ここまで読んでいただいた方はきっとマイクロサービスについて高い関心を持っているのだと思います。

NCDCでは、マイクロサービスの導入支援をはじめ、アーキテクチャの設計コンサルティングや新しい技術を取り入れるための技術コンサルティングを行っていますので、お悩みの方はぜひ一度ご相談ください。
NCDCは、AWSの導入とマクロサービスの積極活用でクラウドネイティブ化に取り組んだ事例や、マイ

クロサービスを活用するためのAPI設計標準化の事例など、この分野でも豊富な実績がありますのできっとお役に立てると思います。

関連する記事

システム開発

3大クラウドAWS、Azure、GCPの超簡単比較。 初心者でもわかる特長と選ぶべき理由とは？

Masaharu Harima

公開 : 2020.10.14

システム開発

Kubernetesの基礎から実際に使ってわかったメリット・デメリットまで解説

Keita Ibaraki

公開 : 2020.09.03

レポート

資料公開｜マネージャー層向けモダンアプリケーション開発戦略セミナー

Masaharu Harima

公開 : 2020.05.14

前の記事へ

次の記

お問い合わせ

NCDCのサービスやセミナー依頼などのお問い合わせは
下記のお電話 また、お問い合わせフォームよりお気軽にご連絡ください。

050-3852-6483

お問い合わせフォーム

採用情報
企業情報
お問い合わせ
ナレッジ

NCDC株式会社
Tel. 050-3852-6483　Fax. 03-6636-9576