この記事は最終更新日から5年以上が経過しています。

NewsPicks Advent Calendar 2018 24日目

@hirofumikubo (久保 裕史)

Microservice化を検討する上で考えるべき7つのこと

microservices microservice

最終更新日 2018年12月25日 投稿日 2018年12月24日

はじめまして！

NewsPicks Advent Calendar 2018 の 24日目を担当するNewsPicksエンジニアの久保です。
NewsPicksのAdvent Calendarも残すところあと2日となりました。
ここまでのエントリーはいかがだったでしょうか？NewsPicksエンジニアの色が非常に出た、かなり個性的でおもしろいものになっているのではないかなと思います。

そして今回僕の方では、Microservice化について書いていきたいと思います。
このエントリーを読んだ方が、今後Microservice化を検討する際の参考になればうれしいです。

なぜこのテーマで書こうと思ったか？

この **「Microservice化を検討する上で考えるべき7つのこと」** というテーマで書こうと思った理由は大きく2つです。

1. 今年NewsPicksで、あるサービスの分割を検討するタイミングあり

NewsPicksでは**すでにいくつかのサービスをMicroservice化**し、コアなAPIとは分離されています。
検索系のサービスや別アプリとして提供しているNewsPicks AcademiaのAPI、今年買収した米経済メディアQuartz社のWeb課金サービスなど、が分割されたサービスになります。

そして今年 **最もコアなAPI** を切り崩し、複数のサービスに分割するべきか検討するタイミングがありました。
そのときの僕個人がMicroservice化を検討する上でどのようなことを考慮したか、を中心に紹介できればと思いました。

2. SRE LoungeでMicroservice化の苦悩を吐露

プライベートで、私や@katsuhisa__さんを中心に **SRE Lounge** というSREチームを持つ企業がその取り組みを紹介する勉強会を運営しています。
（※前回のSRE Loungeはこちら）

SREとは、Site Reliability Engineeringの略でGoogleが提唱したシステム管理とサービス運用を担うチームのことです。

このSRE Loungeにて、いくつかの企業がMicroservice化の苦悩について語る場面があり、今後Microservice化を検討する方がMicroservice化の功罪を理解した上で導入の意思決定ができるような手助けができればと思いました。

導入ではなく、検討

上記2つの理由から、**Microservice化を検討する上で考えるべきこと**について紹介できればと思います。
あくまでも**検討段階**で考えることであって、Microservice化を**導入する上で考えるべきことではない**ので、ご理解ください！

Microservice化のメリット

Microservice化のメリットとしては下記とされています。

- あるサービスの障害が隔離され、サービス全体としては縮退した状態で稼働できる
- 分割されたサービス単位でのスケーリング（水平・垂直ともに）が可能
- サービスと組織構造を一致させることで、チームごとに1つのことに集中でき、生産性やベロシティ向上
- リリース時の影響範囲が小さくなり、リリースのハードル低下。デプロイが容易になる。

しかし、これらのメリットだけを理由に、安易にサービスを分割しMicroservice化することはおすすめしません。
ではなぜおすすめしないのか？ Microservice化を検討する上でどのようなことを考慮する必要があるのか？ を挙げていきたいと思います。

Microservice化を検討する上で考えるべきこととは？

1.どこを境界にシステムを分割するのか？

「エリック・エヴァンスのドメイン駆動設計」の中で出てくる、**Bounded Context（コンテキスト境界）**という概念がMicroservice化を考える上で重要になってきます。
どこを境界線にシステムを分割するのか、つまりどの単位でMicroservice化しシステムを複数のサービスで分割するのか、です。

この分割の境界を見誤ると、重複した振る舞いやコードが生まれたり、サービス間の連携が複雑になってしまうためです。
プロダクトが0の新規開発段階からMicroservice化することがアンチパターンとされている理由も、ドメインの理解が低い段階では、**適切な境界線を定義できない**可能性が高いとされているからです。
このコンテキスト境界の設定に失敗し、結果的にモノリシックに戻すとしても膨大なコストがかかるので、分割の境界線は慎重に判断したいですね。

2.複雑さとのトレードオフ

Microservice化のメリットで挙げたように、分割することでモノリシックなアプリケーションはスリムになり、シンプルになります。
しかし、システム全体で捉えると、分割すればするほど **複雑** になっていきます。

システム監視の対象は増え、障害時にはどのサービスで問題が発生しているかのトレーシングは複雑になり、ローカル開発時には必要なサービスを必要な数だけ立ち上げる必要があり...

とシステム全体で捉えると、シンプルの真逆の複雑の一途を辿ります。
分割することで生まれる複雑さ < Microservice化のメリット かどうかを見極めるのが重要になってきます。

3.システム全体の設計変更やマイクロサービスエコシステムの構築が必要

単なるアプリケーションの分割だけではない、システム全体を含めた設計変更や、その変更される設計にあった環境の構築が必要になります。

2.複雑さとのトレードオフで述べたように、複雑になっていくシステムに合わせた環境の整備が必要になります。
これを、プロダクションレディマイクロサービスでは、**マイクロサービスエコシステム**と定義しています。

アプリケーションは分割したが、アプリケーション間の連携をテストできる環境がなかったり、トレーシングシステムを導入していないことで障害時の復旧に時間がかかったり、など単純なアプリケーションの分割以外の環境の構築と運用のコストがかかることの認識が必要です。

NewsPicksでもまさに、このエコシステムの整備に課題があり、絶賛改善しているところではあります。
開発するメンバーや開発案件が増え、これまでのアーキテクチャではテストする環境が枯渇する問題が深刻になってきたため、Microserviceに合った新しいアーキテクチャのステージング環境や開発環境を準備し直している最中です。

4.技術的スプロールの発生

技術的スプロールとは、Microservice化によって、サービスの数、利用ライブラリの種類、CI/CD方式、テスト手法、アラート/監視システムの種類、開発言語数、ツールなどが、チームの数だけ存在し、膨大に膨れ上がる状態のことです。

いるという問題に、多くの企業が共感していました。

Microservice化によって、開発する機能やサービスの性質にあった技術選定が可能になりますが、組織で統一した**技術選定の方針や標準化**が必要です。

5.「コンウェイの法則」に基づく組織設計の懸念点

「コンウェイの法則」に基づく組織設計が必要なのはいうまでもないですが、組織を再編成できるほどのリソースがあるかは見極めが必要に思います。

モノリシックの場合、Opsを担当するチームは1つだけであるため、オンコール対応のローテーションを回すことはそれほど難しくないかもしれませんが、Microservice化によってサービスが分割され、チームごとにOpsを担う場合に、ローテーションを回すことが可能な体制かどうかは重要です。
仮に、運用担当者だけでなく開発者も含めてローテーションしたとしても、今度は開発のベロシティが下がってしまう危険性があります。

6.「逆コンウェイの法則」の罠

「コンウェイの法則」の逆も成り立つと言われています。その場合、小さく、独立したチームが複数生まれることになると思います。つまり、開発組織や開発者もMicroserviceに似てくるということになります。
その結果、開発チーム間のコミュニケーション不足が生まれ、知識の共有が薄れ、組織のサイロ化が進みます。

この問題を緩和するためには、mercariさんのような、 **サービス単位でチームを構成し、かつ開発から運用まで担う** 仕組みがマッチしているように思います。

NewsPicksでも、クライアントサイドやサーバサイド、インフラ、運用のような分け方ではなく、 **サービス単位でチームが組織され、クライアントから運用までを担う構成**になっています。
加えて、月に1回の全体ミーティングの場を **コミュニケーションの場** として位置づけ、サイロ化にならないように積極的に取り組んでいます。

ですので、いまのエンジニアリソースで、サービス単位で、かつクライアント〜運用を担う構成に編成できるかどうか判断することは重要です。

7.技術的に、だけではなくユーザへの価値もUpdateされるか

Microservice化によって、技術的に効率化され改善されることは当然ですが、 **ユーザへ提供する価値がどう変わるのか** は最も重要に思います。

これまで述べてきたように、エコシステムの構築などMicroservice化には莫大なコストがかかります。
その間、Microservice化を担当するチームの開発リソースを新規機能開発や機能改修などには割けられず、下手をすればプロダクトや会社の成長を妨げる危険性もあります。

ですので、いまこのタイミングでMicroservice化を推し進めることが、 **短期的・中長期的にみて、ユーザへ提供する価値がどう変化するのか、プロダクトや会社の成長にどう影響を与えるのか** 、という視点は非常に大事に思います。

NewsPicksでもここは非常に大事にしており、NewsPicksでは **ユーザへ提供する価値が大きく変わる開発と併せて** 、技術的にもUpdateさせることが多いです。
UIリニューアルや新機能リリース、などユーザへ提供する価値が変わるタイミングで一緒に行うことで何のためにMicroservice化するか、何のために開発するかの目的を見失わずに済みます。

まとめ

僕個人としては、Microservice化には全く否定的ではありません。
システムの規模が大きくなればなるほど、モノリシックなアプリケーションでは無理があり、Microservice化が求められると思います。

しかし、安直な判断によるMicroservice化には反対です。
SRE Loungeを通して様々な企業の取り組みを聞く中で、やはりMicroservice化は非常に難易度が高いと感じています。だからこそ、Microservice化によるメリットを最大限得られるようにしっかりと考慮された上でMicroservice化が進められるのがベストだと思います。

参考文献

- マイクロサービスアーキテクチャ （Sam Newman著）

- Goodbye Microservices: From 100s of problem children to 1 superstar
- メルカリは開発組織を拡大するためにマイクロサービスアーキテクチャを採用した（前編）。Mercari Tech Conf 2018
- メルカリは開発組織を拡大するためにマイクロサービスアーキテクチャを採用した（後編）。Mercari Tech Conf 2018
- microservicesに分割する際に注意するべき5つのこと
- MicroservicePrerequisites

最後に

NewsPicksは5年後、世界で最も影響力のある経済メディアになるという目標を実現するための仲間を募集しています。他の Advent Calendar の記事もご覧いただけると、NewsPicksの事業領域、社風、カルチャーなどもわかると思います。興味をお持ちいただいた方はぜひこちらからご連絡ください!!

Advent Calendarの最終日の明日(12/25)の担当はNewsPicks ブランドデザインチームのチームリーダの木下（@watarukino）です。お楽しみに！

追記

NewsPicks Advent Calendarの最終日の記事は、【千秋楽】2018年の大相撲をエンジニア視点で振り返る です！



0



99
78

関連記事 Recommended by

マイクロサービスの粒度
by soichiro0311

Consumer-Driven Contracts testingを徹底解説！
by AHA_oretama

『Microservice Patterns』 まとめ
by yasuabe2613

KubernetesとNode.jsでマイクロサービスを作成する 1/6 概要
by rairaiso

コメント

この記事にコメントはありません。

T コメントする

プレビュー

コミュニティガイドラインに基づき、良識ある内容を心がけましょう。

テキストを入力

0B / 100MB 投稿する

99 78