# Azure でのマイクロサービス事情

## ～ Azure Service Fabric が必要な理由 ～

サイオステクノロジー株式会社

アプリケーションコンサルティンググループ

水倉　良明




2018/2/22

# 自己紹介

サイオステクノロジー株式会社

技術部　アプリケーションコンサルティンググループ

シニアアーキテクト

水倉 良明


Powered by https://wordart.com/

- クラウド基盤中心にインフラ構築からアプリ開発まで
  - データ分析基盤構築
  - Webアプリ開発
  - API Management
- 片道２時間通勤のため、週２～３日のリモートワークに助けられてます

# Agenda

- マイクロサービスのおさらい
  - メリット
  - マイクロサービスの難しさ
- Azure Service Fabricとは？
  - 概念
  - アーキテクチャ
  - プログラミングモデル
  - 注意点
  - クラスタ作成イメージ
  - デモ
- まとめ

# マイクロサービスのおさらい

# マイクロサービスのはじまり

- Martin Fowler氏らが2014年に発表した記事により広まったと言われる
  - “Microservices”
    ~a definition of this new architectural term
    - https://martinfowler.com/articles/microservices.html

- 小回りの利かないモノリシック（一枚岩）なシステムから疎結合なサービスの集合体へ

# マイクロサービスが注目されている背景

- 目まぐるしい環境変化に即対応することが求められている

Agilityを実現するためのアーキテクチャとして注目

影響範囲の最小化、より細かい粒度でのリリースサイクルを実現して、迅速な開発・運用を実現する

# マイクロサービスのメリット

- サービス毎に独立した開発・リリースサイクルで迅速な機能改修
- 性能とコストの最適化
  - サービス（機能）単位でのスケールアップ/アウト
  - 「１VM１アプリ」から「１VM複数サービス」による高密度なコンピューティング
- テクノロジーの多様性
  - サービス毎に最適な言語、テクノロジーの採用
- 小規模なチームによるスピード感のある開発・運用の実現
  - DevOpsとの相性の良さ
- 障害の分離
  - 機能・リソースの分離による障害影響範囲の最小化

# 理想と現実のギャップ

巨大なモノリスが小さく単純なサービスになる一方・・・

1. マイクロサービス化することが難しい

2. マイクロサービス化できたとしても運用が難しい

# マイクロサービスの難しさ

- まず、マイクロサービス化することが難しい
- 適切なサービス単位、データ単位の見極め
  - ドメイン分析スキルが必要（Domain-Driven Designなアプローチ）
- 疎結合とはいえ、サービス間の依存関係をゼロにできるわけではない
  - 依存関係を踏まえた設計が必要
- マイクロサービス 化することが目的にならないように・・・
  - マイクロサービス化によって得られるもの、失うものの見極めが必要
  - 何のために取り組むのかを見失ってはいけない
  - アプリケーションによってはデメリットになることも
    - “Enough with the microservices”
      https://aadrake.com/posts/2017-05-20-enough-with-the-microservices.html

# マイクロサービスの難しさ

- マイクロサービス化できたとしても運用が難しい
- 分散システムはそもそも複雑なもの
  - 大量のサービス管理
    - １つのデプロイではなく、数十、数百ものサービスのデプロイへ
    - 各サービスのビルド、テスト、デプロイ、シームレスな連携
    - 各サービスのリソース確保
  - ネットワークの輻輳と遅延
    - モノリスではインプロセスの高速な処理
    - マイクロサービスではアプリへの１つのリクエストがサービスを連鎖的に呼び出す多数の内部リクエストになるケースがある

# マイクロサービスの難しさ

- データの整合性
  - モノリス
    - 全てのデータが単一のデータベースに格納される
    - 参照整合性の確保。厳格なトランザクション制御が可能
  - マイクロサービス
    - 複数のデータベースに分散
    - サービス間のデータの完全性を担保することの難しさ
    - レプリケーションのタイムラグへの対処

# マイクロサービスの難しさ

- テスト
  - サービスの依存関係によりテストが複雑になる
  - 複数のサービスが異なるペースでバージョンアップされる中で、一貫した環境を再現することの難しさ
  - これまで以上に環境への考慮や、自動化への投資が必要
    - ブルーグリーンデプロイメント
    - カナリアリリース
    - ダークカナリアリリース
    - A/Bテスト
- 組織文化
  - DevOpsが定着していないと安定した運用が困難

# マイクロサービスの難しさ

- 乱立するミドルウェア、周辺ツール群

# マイクロサービスの難しさにどう挑むか

# Azure Service Fabricとは？

# Azure Service Fabric

SIOS

## マイクロサービス時代のPaaS
## 分散システムの複雑さを高度に抽象化

# Azure Service Fabricとは？

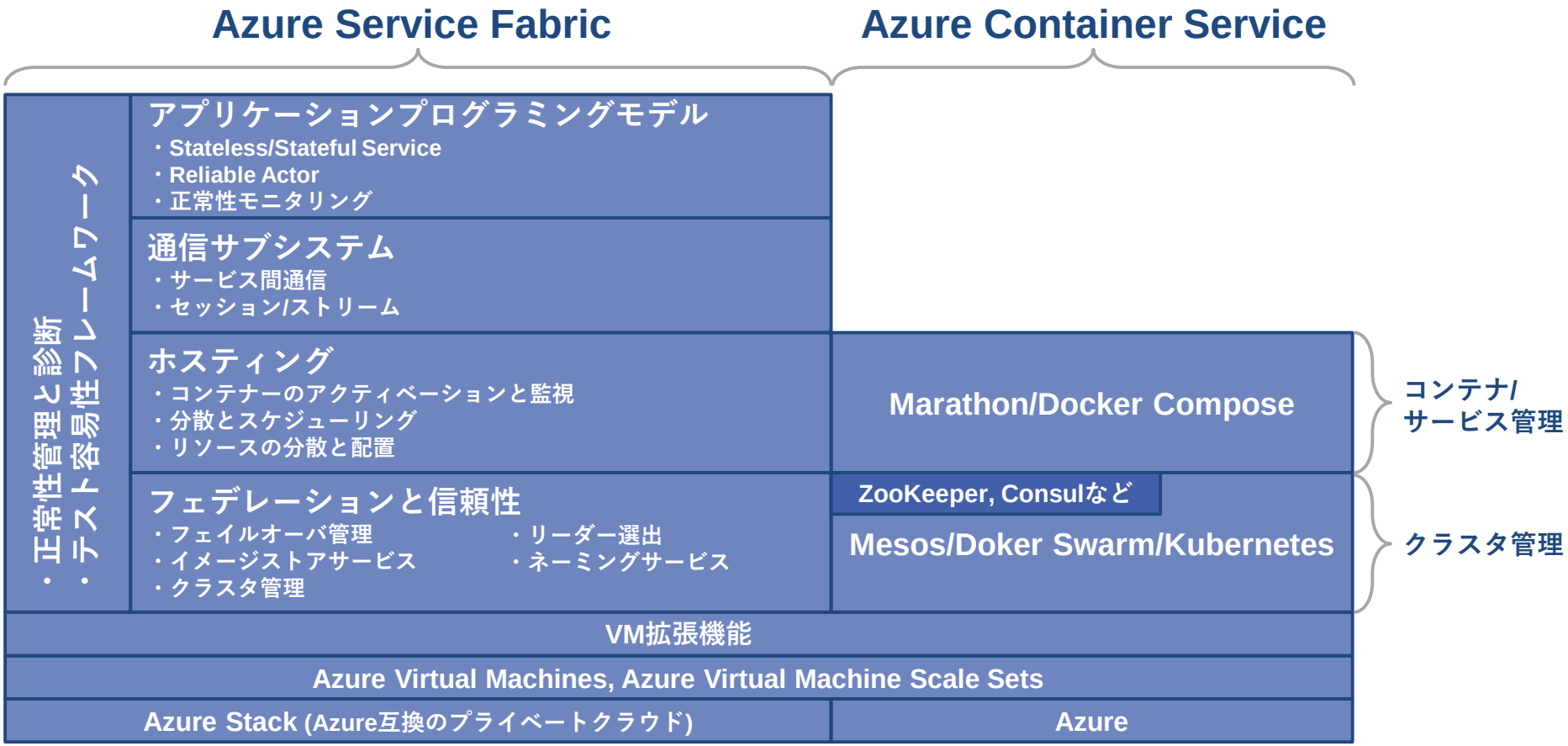
- GA開始は2016年3月だが、７年以上のサービス稼働実績
  - Azureコアインフラ
  - Azure SQL Database
  - Azure DocumentDB
  - Azure Event Hubs
  - Cortana
  - Bing
  - Skype for Business
  - Power BI

**Microsoft社がAzureで培った分散システム技術をサービスとして一般提供したものと言える**

# Azure Service Fabricとは？

- フルマネージドでフルスタックなマイクロサービスフレームワーク
  - クラスタ上でサービスを実行、管理するための分散システム基盤
  - アプリケーションレイヤの各種プログラミングモデルも提供

# Azure Service Fabricアーキテクチャ






出典：「マイクロサービス with Docker on Azure」Boris Scholl他 (著)

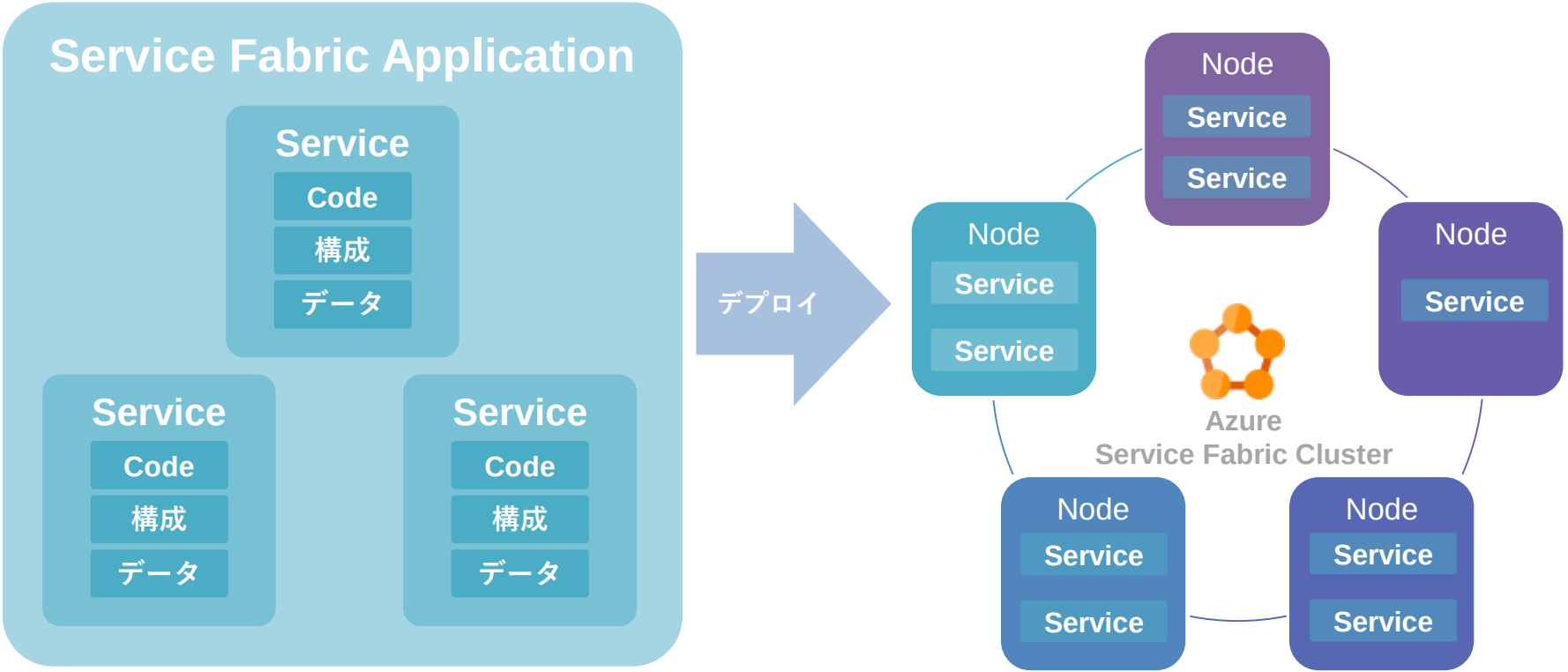
# Azure Service Fabricクラスタはどこでも動く

- Azure
- Azure以外のパブリッククラウド
- オンプレミス
- OSはWindows Server、Linuxのどちらでも動作
- ローカルでも動作。開発環境構築も容易
  - Azure Service Fabric SDK (※) がローカルクラスタを提供

(※) 2017年3月にOSS化

# Azure Service Fabricアプリの論理構造

- 各サービスはどのホストに配置されるかを原則意識しない（※あえて意識することも可能。後述）

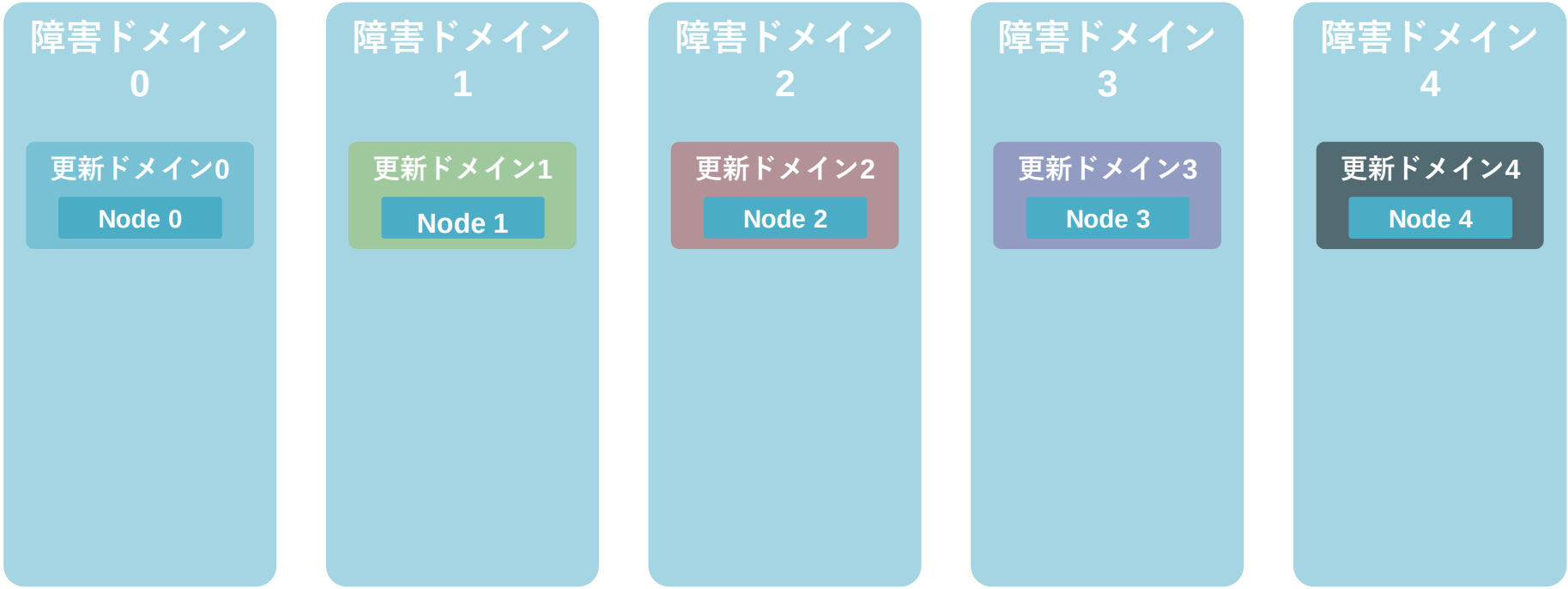


# 障害/更新ドメインを考慮したサービス分散

- 障害ドメイン
  - 物理的な障害単位。電源、冷却装置、ネットワークインフラを共有したサーバーリソースの集まり
- 更新ドメイン
  - 論理的な更新単位
    - 同じ更新ドメインに所属しているサービスインスタンスは同時に停止・更新される
    - 更新ドメインが別であれば同時に停止・更新されることがない

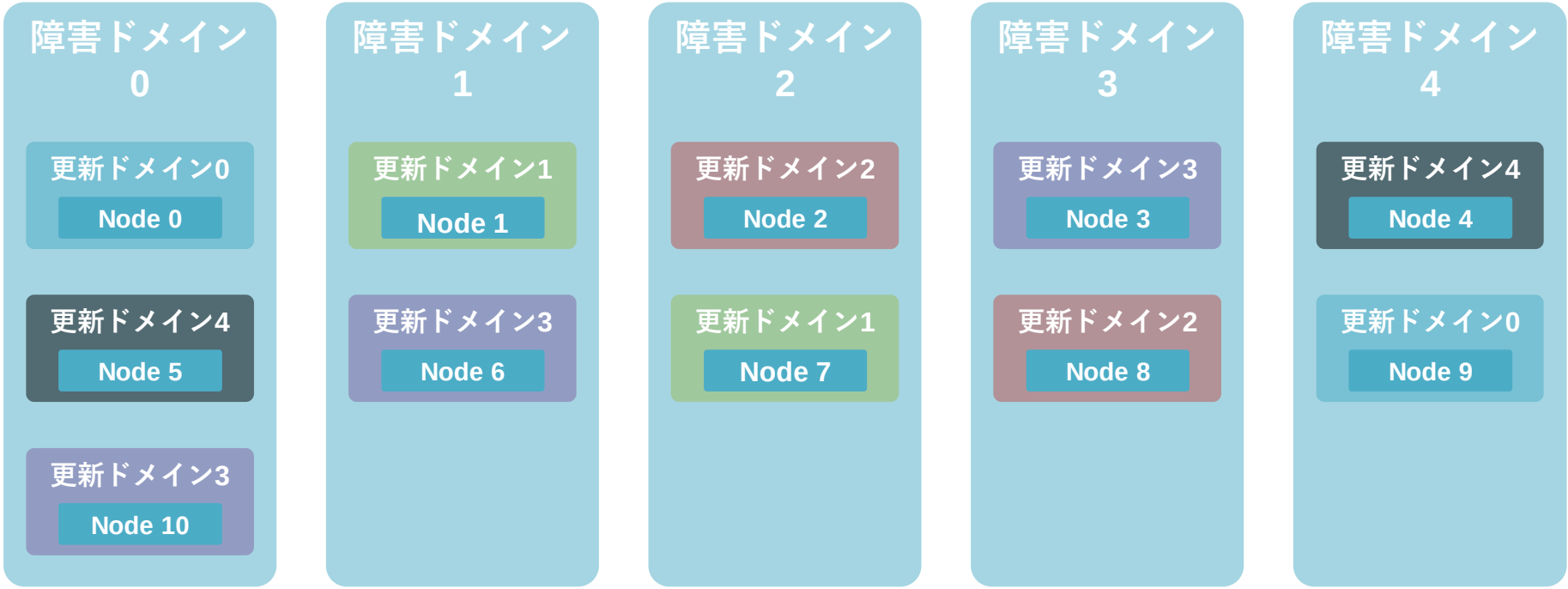
# 障害/更新ドメインを考慮したノード配置

- 最低３つ以上の障害ドメインに分散される
- 既定では５ノードのクラスタは５つの障害ドメインと5つの更新ドメインに分散される



# 障害/更新ドメインを考慮したノード配置

- 障害ドメインと更新ドメインが一意な組合せになるようにノードが配置されていく
- 配置条件を指定することも可能（他よりもリソースを必要とするサービスは一定のスペック以上のノードに配置など）



# Azure Service Fabricでサポートされるservice

- Stateless Service
  - ローカルに状態を保持しない
    - 状態保持不要、または外部に状態を保持
  - 分散システムで扱いやすい
- Stateful Service
  - ローカルに状態を保持する
  - データの同期制御、データ完全性を保証することの難しさがある
    → ASFがカバーしてくれる

# Azure Service Fabricプログラミングモデル

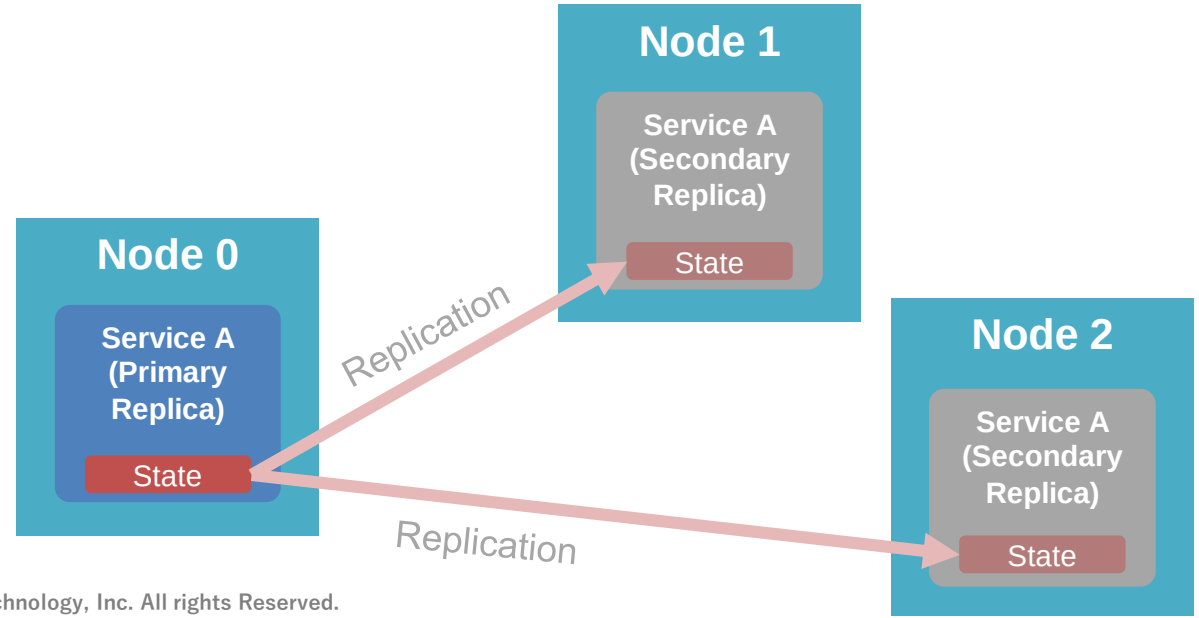
- Reliable Service
- Reliable Actor
- ASP.NET Core
- コンテナ
- ゲスト実行可能ファイル

# PGモデル：Reliable Service – Stateless Service

| メリット | デメリット |
| --- | --- |
| • ASFの全機能の恩恵を受けることができる<br>• Pluggableな通信プロトコル。複数の通信プロトコルでListenすることも可能。<br>  • HTTP<br>  • WebSocket<br>  • WCF<br>  • カスタムTCPプロトコル | • 状態を外部へ永続化している場合、状態へのアクセスがステートフルサービスよりも遅くなる |

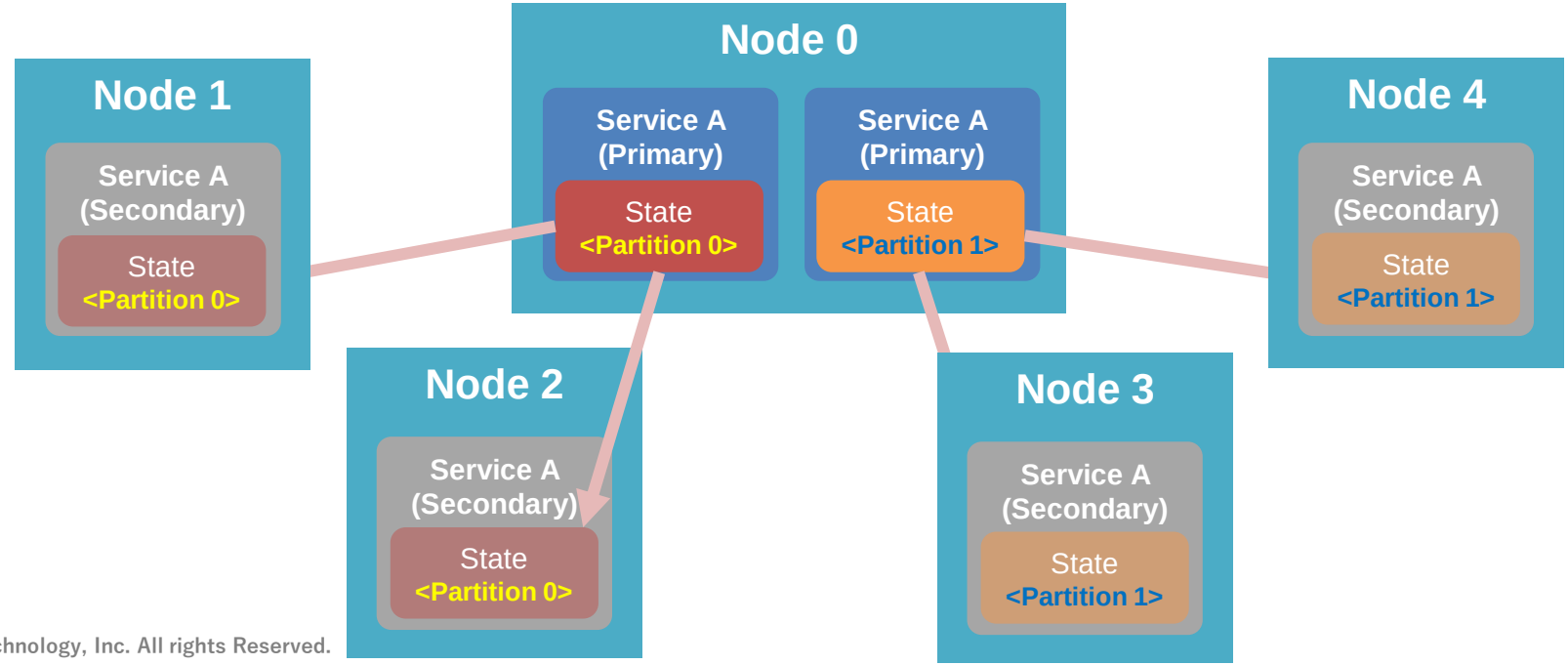
# PGモデル：Reliable Service – Stateful Service

- 状態はレプリケートされ、強い整合性が保証される（トランザクションサポート）
  - 開発者はサービスに保持したい情報をReliableColletionに保存するだけで自動制御される
- Primary Replicaダウン時はSecondaryがPrimaryに昇格する（自動フェイルオーバ）



# PGモデル：Reliable Service – Stateful Service

- レプリケーションするデータを区切るパーティションもサポート
- Primary Replicaをパーティション毎に保持できることで、Primaryがボトルネックにならない



# PGモデル：Reliable Service – Stateful Service

| メリット | デメリット |
| --- | --- |
| • Stateless Serviceと同様のメリット<br>• 状態をローカルに保持するため、低レイテンシで動作<br>• 開発者は状態のレプリケーション制御を意識しないでよい<br>• パーティション機能によって、ステートフルでありながらも容易にスケールアウト可能 | • パーティション数を変更することができない<br>• ノード数に対してパーティション数が少ない場合、ノード数を増やしても性能メリットが得られないので注意<br>  • 例）パーティション数が1000個で、<br>  ノード数が10の場合、<br>  → 1ノード当たり100個のパーティション<br><br>  ノード数を10から100個に増やした場合、<br>  → 1ノード当たり10個のパーティション（性能メリット有り）<br>  ※パーティション数が10個の場合、ノードを増やしてもスケールアウトされない |

# PGモデル：ASP.NET Core

- OSSクラスプラットフォームなWebフレームワーク
  - Web アプリ
  - IoT アプリ
  - モバイル バックエンド
- Reliable Service、ゲスト実行可能ファイルの 2 パターンのホスティングが可能

# PGモデル：ASP.NET Core

| メリット | デメリット |
| --- | --- |
| • Reliable Serviceと同様のメリット<br>• ゲスト実行可能ファイルとしてデプロイした場合、ASF機能は限定的になるものの、既存のASP.NET Coreをコード変更することなく利用可能<br>• ASP.NET経験者はスキルを活用できる | • Webサーバ機能にIIS (HttpSys) とKestrelが利用可能だが、それぞれ考慮すべきことがあることと、Listenerの実装コードに違いが生じるので注意<br>  • HttpSysはステートフルでは非推奨<br>  • Kestrelでは複数プロセスでのポート共有はサポートされない<br>  ※外部公開 or 内部専用、ステートレス or ステートフルで適切な選択をする<br>  https://docs.microsoft.com/ja-jp/azure/service-fabric/service-fabric-reliable-services-communication-aspnetcore |

# PGモデル：Reliable Actor

- Reliable Service上でActorモデルを実装可能なフレームワーク
  - Actorモデル：「すべてはActorである」ととらえて、Actor間のメッセージングを非同期とすることで並行処理を実現する。複雑な分散システムを抽象化できる。
- ASFでは非同期シングルスレッドで動作する
  - ターンベースの並行性
    - 単一のActorインスタンスは同時にメソッドが呼び出されない
    - 複数のActorインスタンスは並行して実行される
  - マルチスレッドプログラミングの複雑さを抑えつつ、並行性を確保

# PGモデル：Reliable Actor

| メリット | デメリット |
| --- | --- |
| • ターンベースの並行性により、非同期並行処理を非常に容易に実装できる<br>• Reliable Serviceがベースなため、Stateless/ful共に対応。ASF提供機能を全て利用できる | • 高度に抽象化されているとはいえ、デバッグやパフォーマンスチューニングをするには内部の仕組みを把握する必要がある（これに限った話ではありませんが・・・） |

# プログラミングモデル：コンテナ

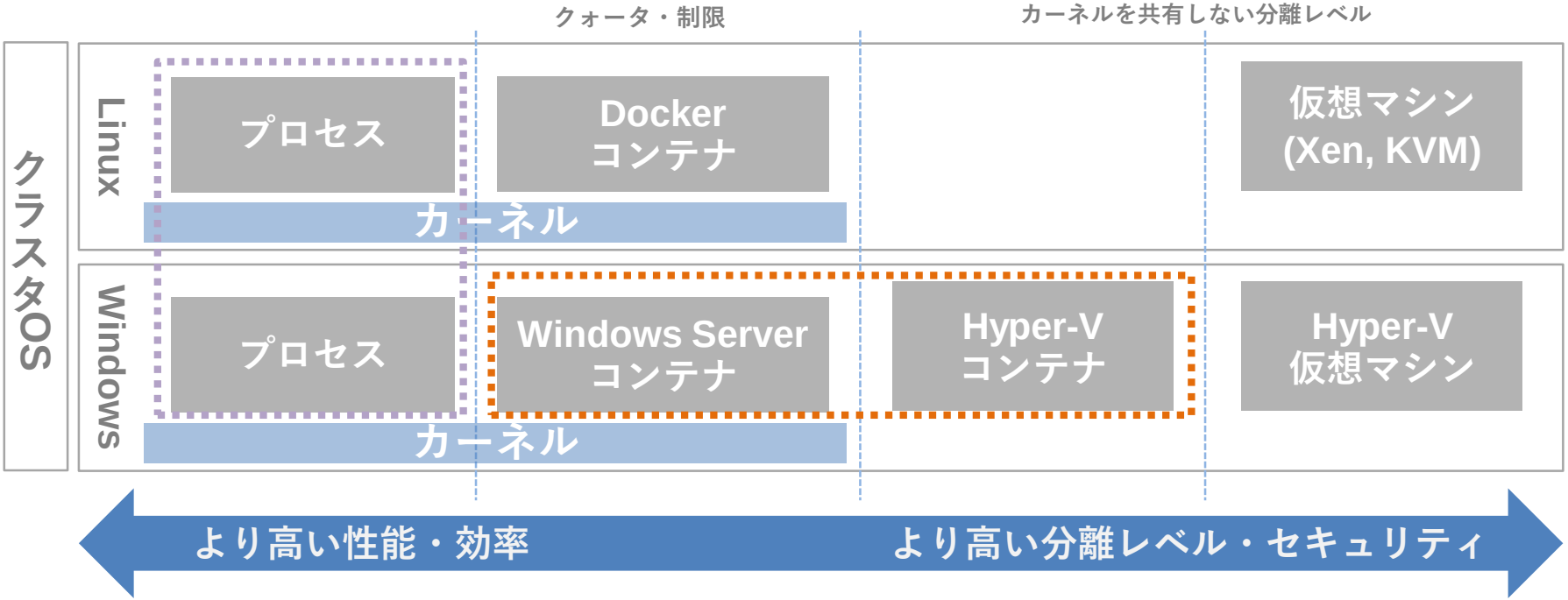
- コンテナ
  - 各種プログラミングモデルをコンテナ内のサービスとしてデプロイ
    - 既存アプリ（ゲスト実行可能ファイル）
    - Stateless/Stateful Reliable Service
    - Reliable Actor
  - Linux/Windowsコンテナのデプロイをサポート
  - プロセス内のサービス（＝コンテナ以外のASFサービス）とコンテナ内のサービスを同じアプリケーション内で共存可能

# PGモデル：コンテナ

| メリット | デメリット |
| --- | --- |
| • 既存のASP.NET MVCアプリをASP.NET Coreに移行せずに使い続けることができる<br>• 既存のコンテナイメージを活用できる<br>  • 例）より多くのバックエンド処理に対応するためにWebフロントエンドにNGINXコンテナを配置する<br>• 同一ノード内の他サービスの影響を軽減できる<br>  • コンテナのリソース制限機能を活用することで、「Noisy Neighbours」問題へ対処<br>• 同一アプリケーション内で他ASFサービスとコンテナ内のサービスを共存できる | • 高速な軽量コンテナではあるが、ASFデフォルトのプロセス実行されるサービスよりはアクティブ化やI/Oが若干ボトルネックになる |

# コンテナの種類とサポートされる環境

- Windowsは分離レベルの異なる２つのコンテナを提供
- コンテナ以外のプログラミングモデルではサービスはプロセスとして実行される




https://docs.microsoft.com/ja-jp/azure/service-fabric/service-fabric-containers-overview#container-types-and-supported-environments

# PGモデル：ゲスト実行可能ファイル

- 既存の任意言語実装のプログラムをサービスとして実行する
  - C++, Java, Node.js, PHP, etc.
- コードレベルでASFのバイナリを一切参照しないカスタムアプリケーション

| メリット | デメリット |
| --- | --- |
| • 幅広い言語に対応しているため、既存資産が活用できる | • 恩恵を受けられるASF PaaS機能が限られている<br>  ◦ 可用性の向上<br>  ◦ 正常性の監視<br>  ◦ ライフサイクル管理<br>  ◦ 高密度なコンピューティング<br>  ◦ クラスタ内の他サービスへのアクセス（※ASF REST API呼出が必要）<br>• ステートフルサービスに対応していない<br>  ◦ 既存アプリがステートを保持している場合、手当てが必要 |

# PGモデルとStateless/Statefulの対応状況

| プログラミングモデル | Stateless | Stateful | 備考 |
|---|---|---|---|
| Reliable Service | ● | ● | |
| Reliable Actor | ● | ● | |
| .NET Core | ● | ● | ゲスト実行可能ファイルとして配置した時はそちらに準ずる |
| **コンテナ** | - | - | コンテナ内のPGモデルに準ずる |
| **ゲスト実行可能ファイル** | ● | X | |

# クラスタOSと言語

| クラスタOS | ゲスト実行可能ファイル | Reliable Service<br>Reliable Actor | コンテナ |
|---|---|---|---|
| **Windows** | Windows | .NET Core | Windows |
| **Linux** | Linux | .NET Core<br>Java | Linux |

# Azure Service Fabric開発ツール

| OS | 開発ツール |
| --- | --- |
| Windows | Visual Studio<br>PowerShell |
| Linux | Eclipse<br>Yoeman |

# Azure Service Fabricサービスの監視

- Azureサービスと統合されている

| レイヤ | 監視手段 |
| --- | --- |
| クラスタ | Azure Monitor<br>Log Analytics |
| アプリケーション | Application Insights |

# Azure Service Fabric注意点（2018/2/19時点）

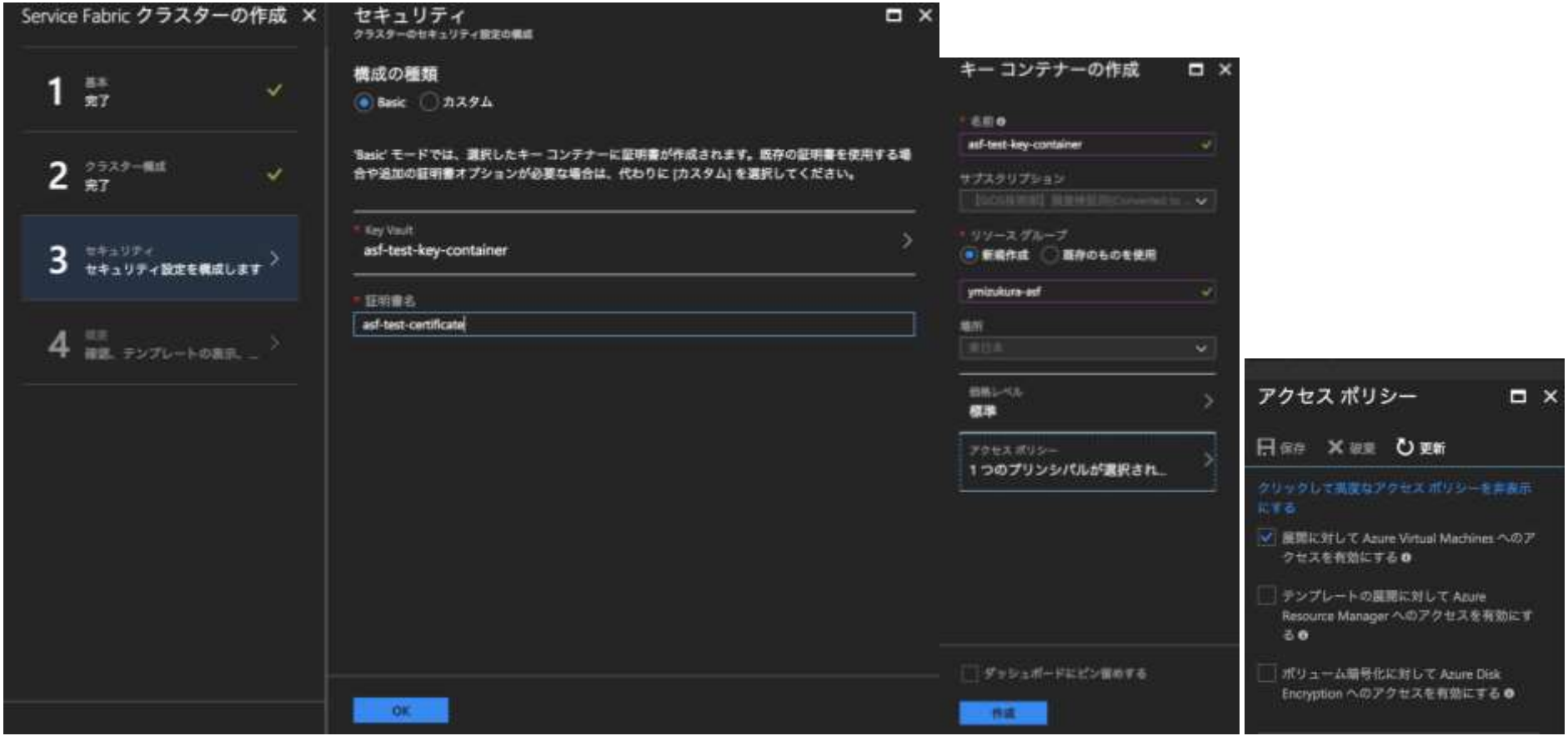
- ASFフル機能を利用できるReliable Service/ActorのJavaはWindowsに対応していない（Linux版Eclipseのみ対応）
- 特定OSでは未サポートであったり、プレビュー段階のものがあるので、対応状況は要確認
  - Windows 10での Service Fabricクラスターへのコンテナのデプロイは未サポート
  - プレビュー段階
    - Docker Compose
    - Windows Server バージョン 1709はプレビュー版。Open ネットワークおよび Service FabricのDNS サービスは機能しない。

# Azure Service Fabricクラスターの作成イメージ

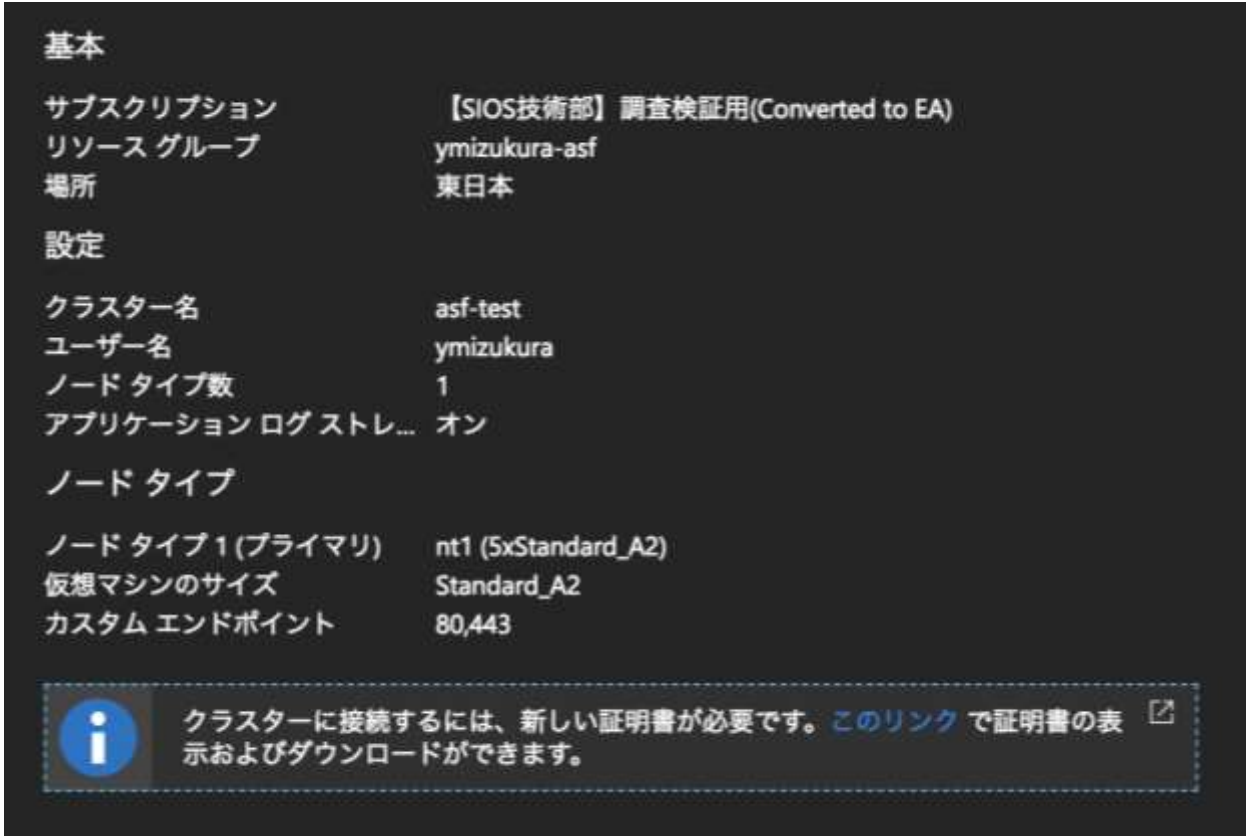
# Azure Service Fabricクラスターの作成イメージ

# Azure Service Fabricクラスターの作成イメージ

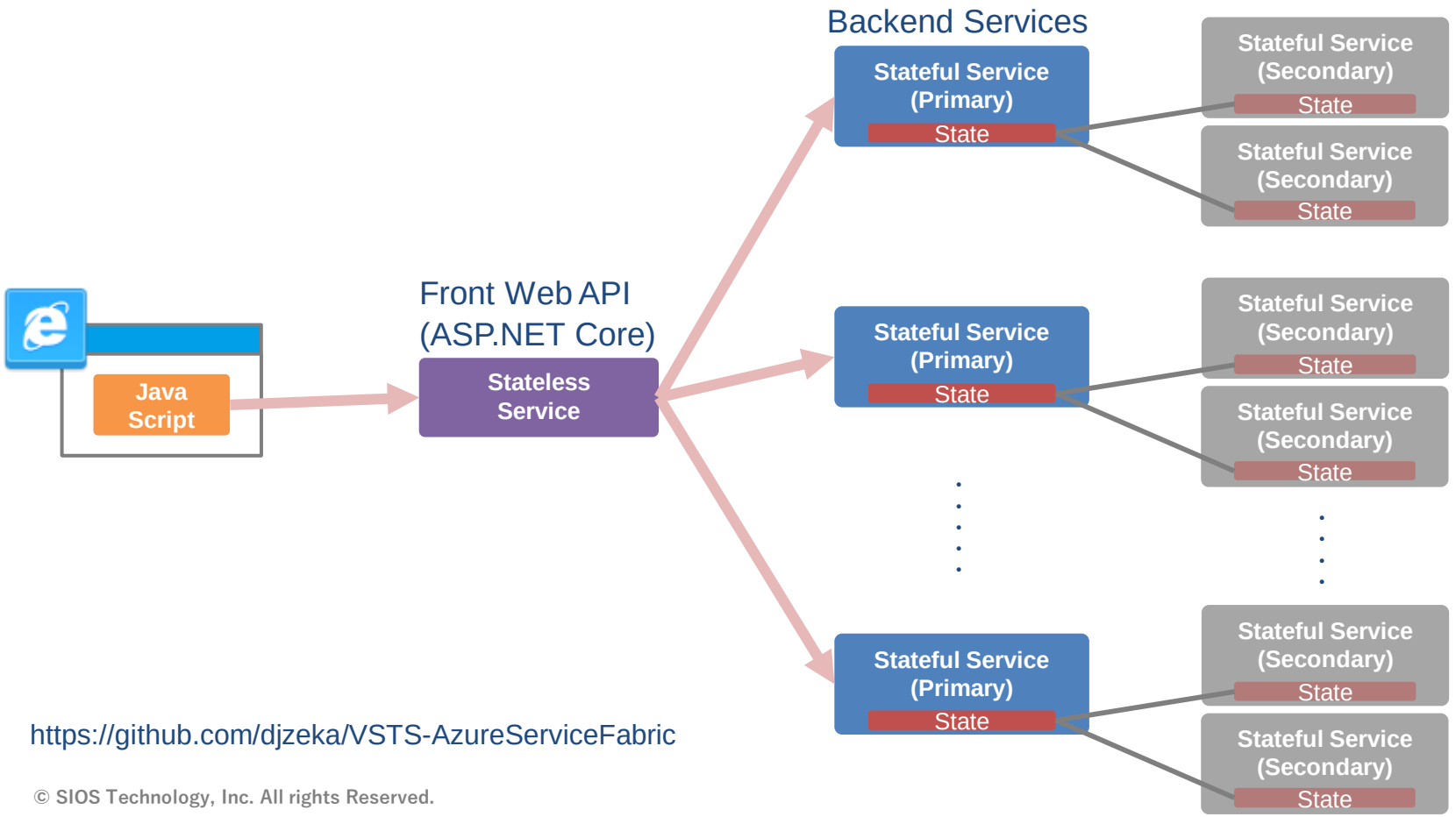

# Azure Service Fabricクラスターの作成イメージ



# Azure Service Fabricクラスターの作成イメージ



# デモ




https://github.com/djzeka/VSTS-AzureServiceFabric

# まとめ

- マイクロサービスは難しい
- Azure Service Fabricは、マイクロサービスで考慮すべき“複雑さ”を一定レベルに抑えてくれる
  - マイクロサービスのハードルを下げて、迅速な開発・運用サイクルのメリットを享受できる
  - ビジネスの差別化へリソースを集中できる
- 適切なドメイン分析による適切なサービス単位の見極めは必要

# 参考資料

- Azure Service Fabric概要
  - https://www.slideshare.net/dahatake/azure-service-fabric-69314586
- 「マイクロサービス with Docker on Azure」
  - Boris Scholl他 (著) / 佐藤 直生 (監訳) / クイープ (訳)
  - ISBN：978-4822298845
- 「プログラミングAzure Service Fabric」
  - Haishi Bai (著) / 佐藤 直生 (監訳) / クイープ (訳)
  - ISBN：978-4822298852
- デモ用サンプルコード
  - https://github.com/djzeka/VSTS-AzureServiceFabric

# 最後に・・・

# OSS on Azure技術ブログやっています

## azure.sios.jp


ビギナー向けの基本情報から
現場エンジニアによるディープな技術トピックスまで。
技術を愛する全ての人に。

# ビール片手にDB最新情報を！

3月13日（火）19:00 ～ 品川テラスホール

## Azure DB for PostgreSQL/MySQL 開発マネージャと話そう！

**主催：**
**OSS on Azure 非公式コミュニティ**

- Microsoft Azure上におけるOSS利活用推進のためのビジネス＆技術情報共有コミュニティ、Facebook公開グループです。
- 非公式というのはマイクロソフトさん主導でのコミュニティじゃないよ！という意味合いですが、イベントの度に飲食費用をご支援いただいたりでアレです。

- 来日中の米国マイクロソフト Azure DB for PostgreSQL/MySQL 開発マネージャへの質問大会
- 日本のPostgreSQL/MySQL ユーザ会やその周辺の方々からの最新情報
- 当然ですが、ビール＆軽食ありです。（参加費無料）

https://ossonazure.connpass.com/event/80145/

# ケーキ片手に誕生日パーティを！

2月27日（火）サイオステクノロジーコラボレーションラウンジ

## Japan OSS Promotion Forum 2018 + 20th Party for OSS

### 主催：日本 OSS 推進フォーラム

OSS活用の課題や普及促進をオープンに推進し、産業界全体の発展に寄与することを目的として2004年に発足した民間団体。市場における新たなトレンドなど、最新トピックをテーマとして活動に取り入れ、業界における共通認識の形成・認知の共有、政府提言のとりまとめなどを推進しています。

日本OSS推進フォーラムの年一イベント、来賓講演として経済産業省、基調講演としてプロダクトマネジメント第一人者、AIビジネスの先駆者、各技術部会からの年間活動報告など、そんな真面目なイベントの情報交換会（要参加費）にて・・・

OSS20歳の誕生日パーティ開催！

スペシャルケーキも鋭意制作中！

https://ossforum.connpass.com/event/77003


SIOS