ビジネスの価値向上をもたらす マイクロサービスと レッドハットのソリューション

2018年 7月 10日

レッドハット株式会社
テクニカルセールス本部
ミドルウェアソリューションアーキテクト部
河野　恭之

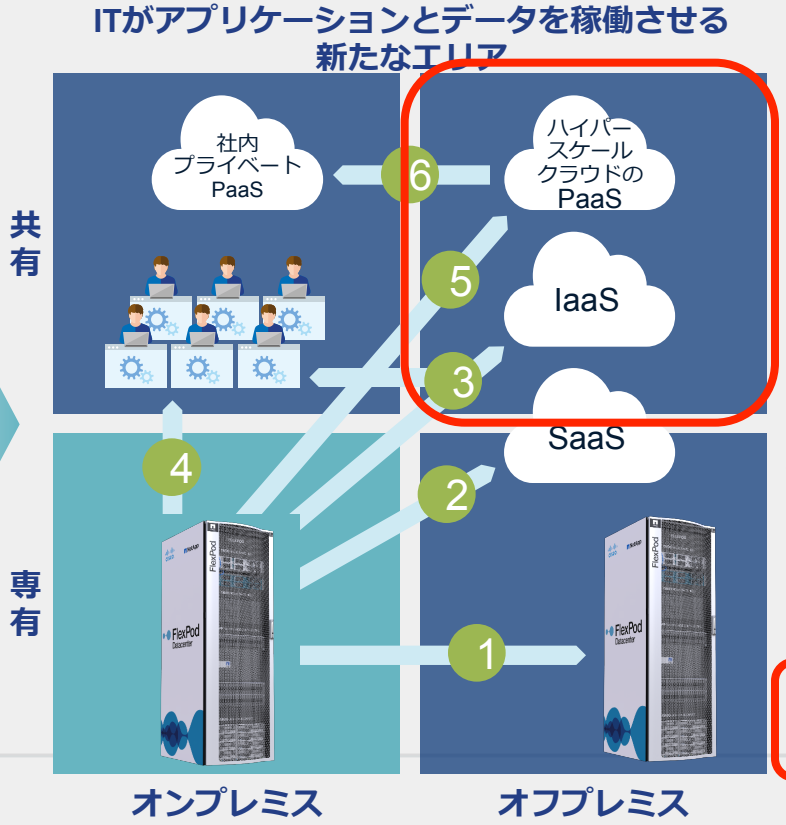
Agenda

- なぜ今、マイクロサービスなのか
- マイクロサービスの開発と運用を実現する
  Red Hat OpenShift Container Platform ソリューション
- マイクロサービスを導入した事例の紹介
- レッドハットのデジタル変革支援メニュー
- まとめ

DXのためのアプリケーションとデータの活用

イノベーションアクセラレーター

- Internet of Things
- コグニティブ/AIシステム
- ロボティクス
- AR / VR
- 3Dプリンティング
- 次世代セキュリティ



1. オンプレミスと同様のインフラをアウトソース
2. SaaSでアプリケーション利用
3. プライベートクラウドとパブリッククラウド間でインフラを移行
4. 社内プライベートクラウドを構築 (IaaS)
5. AWSやAzureなどのクラウドベースのマイクロサービス　基盤の活用 (PaaS)
6. 社内のマイクロサービス基盤への回帰 (PaaS)

なぜ今、マイクロサービスなのか

開発プロセスとITの進化:

SoR
(基幹系)

Development Process	Application Architecture	Deployment & Packaging	Application Infrastructure
Waterfall	Monolithic	Physical Servers	Datacenter
Agile	N-Tier	Virtual Servers	Hosted
DevOps	**Microservices**	**Containers**	**Cloud**

SoE
(新サービス、
差別化領域)

早期リリース
継続的学習/
改善サイクル

独立した単位で
スケールアウト、
改修、テスト

軽量、高集約、ポータビリティ、
運用の簡素化

モノリシックアプリケーションの課題例

デプロイの間隔が長い
　変更の度にアプリケーション全体や密結合の依存が再度テストされ、一度にデプロイする必要がある

起動に時間がかかる
全てのライブラリの読み込みに長い時間がかかり、起動が遅い

スケールしない
　使わない部分も含めて、水平あるいは垂直にシングル・スタックをスケールしなくてはいけない

技術スタックの選択に関する制約
メンバ全員に同じ技術スタックの利用を強いる事になり、最大公約数に至りがち（Javaや.NET、リレーショナル・データベース等）

ダウンしたら全てが終わる
　アプリケーション全体がダウンしてしまう

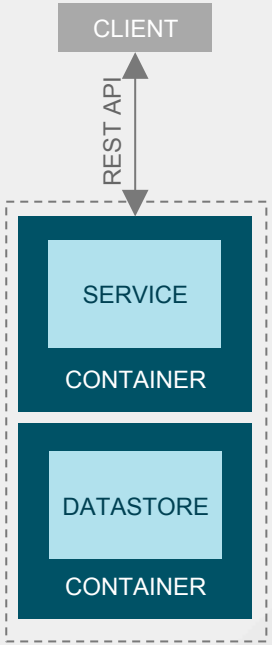
デプロイが大変
アプリケーションが大きくかつ複雑な傾向にあるので、デプロイの難易度が高い。また、いろいろな設定項目や環境変数が多数あり取扱が大変。

APIが後付け
アプリケーション自体がビジネスロジックを扱うため、APIは多くの場合、後で考えられる。社内外の再利用性の低下

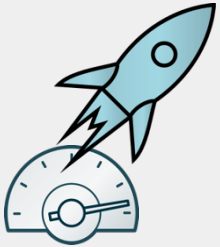
マイクロサービスアーキテクチャ

マイクロサービスアーキテクチャの原則:

- ビジネスドメインを中心としたモデル化
- 自動化の文化 (DevOps)
- 独立してデプロイできる
- 独立してスケールできる
- 独立してリリースできる
- サービスを組み合わせてシステムやアプリケーションを実装
- アンチフラジャイル – プレッシャー下で堅牢性や回復力が向上
- Polyglot (言語やフレームワークに依存しない)
- APIやコントラクトにフォーカス
- 中央集権的でないデータ管理

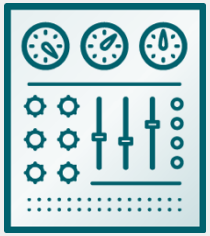


マイクロサービスの価値



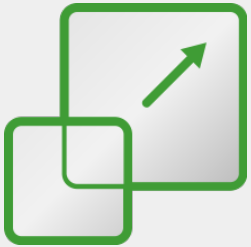
市場投入へのスピード向上

サービスが小さく、自律的であるため、素早く開発してデリバリーできる



運用の効率化

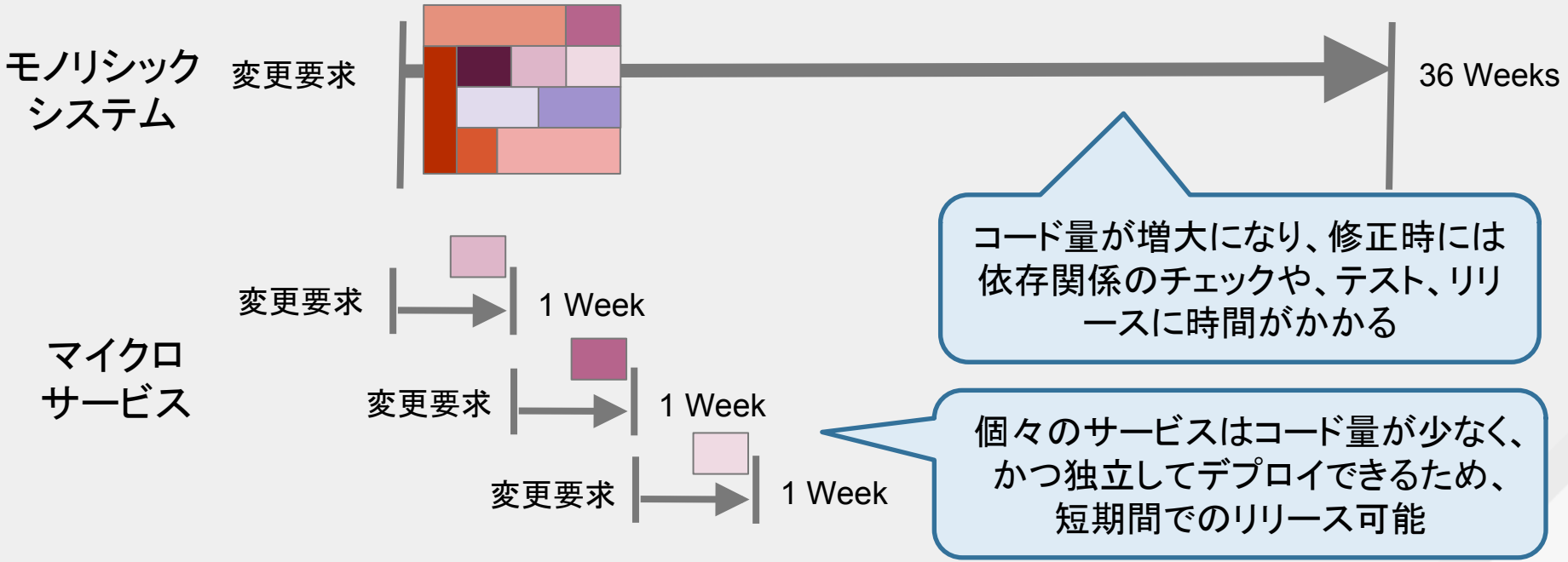
サービスが小さいため、デリバリーの自動化やモニタリングが容易



高拡張性

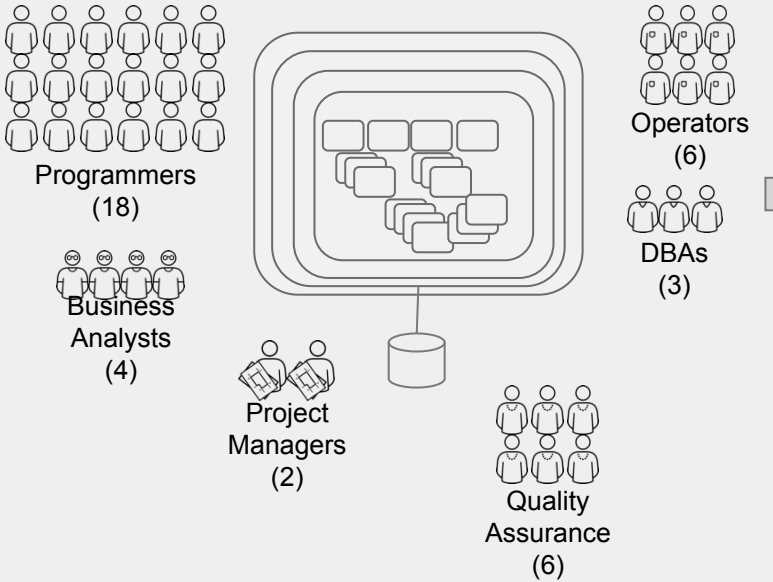
細かい粒度でスケーラビリティを制御でき、リソースの利用を最適化しやすい

マイクロサービスの価値：
市場投入へのスピード向上

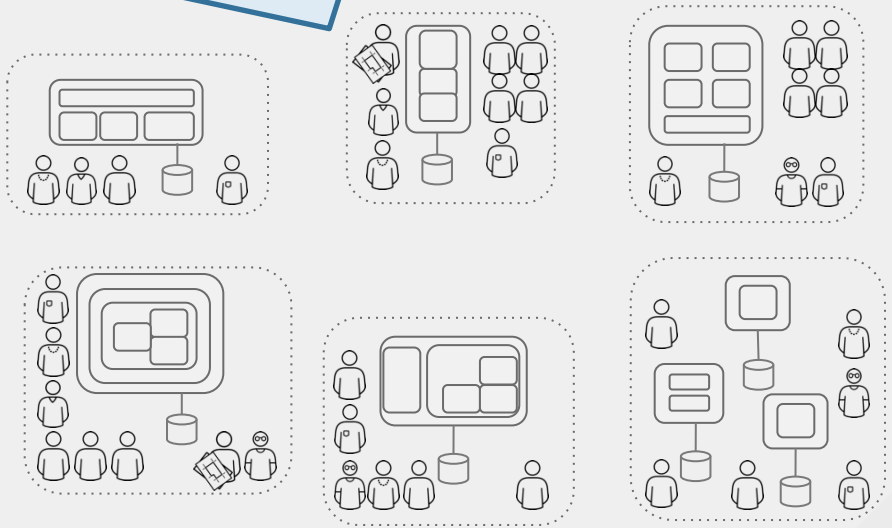


マイクロサービスの価値：運用の効率化

対象範囲が大きいため、プロジェクトチームが大きくなり、役割ごとに細分化されてしまいがち



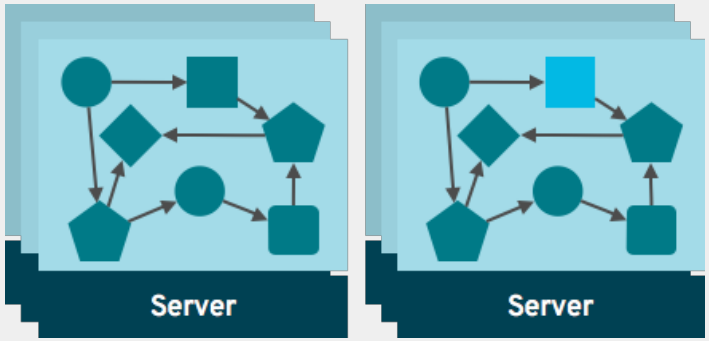
役割ではなくサービスごとにチームを編成することで、自動化や運用管理まで一貫した取組みがしやすい



マイクロサービスの価値：高拡張性

アプリケーション単位でしか
スケールアウトできない

モノリシックなシステム



サービス単位で細かくリソース調整可能

マイクロサービス

マイクロサービス化の適性

マイクロサービスに向くシステムの例

- 複数言語/複数ランタイムを使い分けるシステム(Polyglot)
- 変更が頻繁なシステム
- 大規模なシステム

マイクロサービスに向かない(メリットの少ない)システムの例

- 厳密なACIDトランザクション管理が必要なシステム
- パフォーマンス要件(特にレイテンシ)の厳しいシステム
- 小規模なシステム
- 変更の少ないシステム
- サービス境界が明確になっていないシステム

マイクロサービスの課題

- 開発・構築
  - サービスの境界 / 粒度
  - ビルド / デプロイの効率化
  - サービス間連携 (サービスディスカバリ、オーケストレーション、コレオグラフィー)
  - テスト (サービス単位 / エンドツーエンド)
  - 自律的な回復
- 運用
  - サービスの健全性の管理 / 監視
  - サービス障害対応 (ログ集中管理、トレーサビリティ)
  - サービス間の依存関係の管理 (バージョニング、データマイグレーション)
  - パフォーマンス管理 / リソース管理
  - API管理 (特に外部公開用)
- セキュリティ
  - 分散環境での認証 / 認可
  - 暗号化
  - 監査

マイクロサービスの開発と運用を実現する Red Hat OpenShift Container Platform

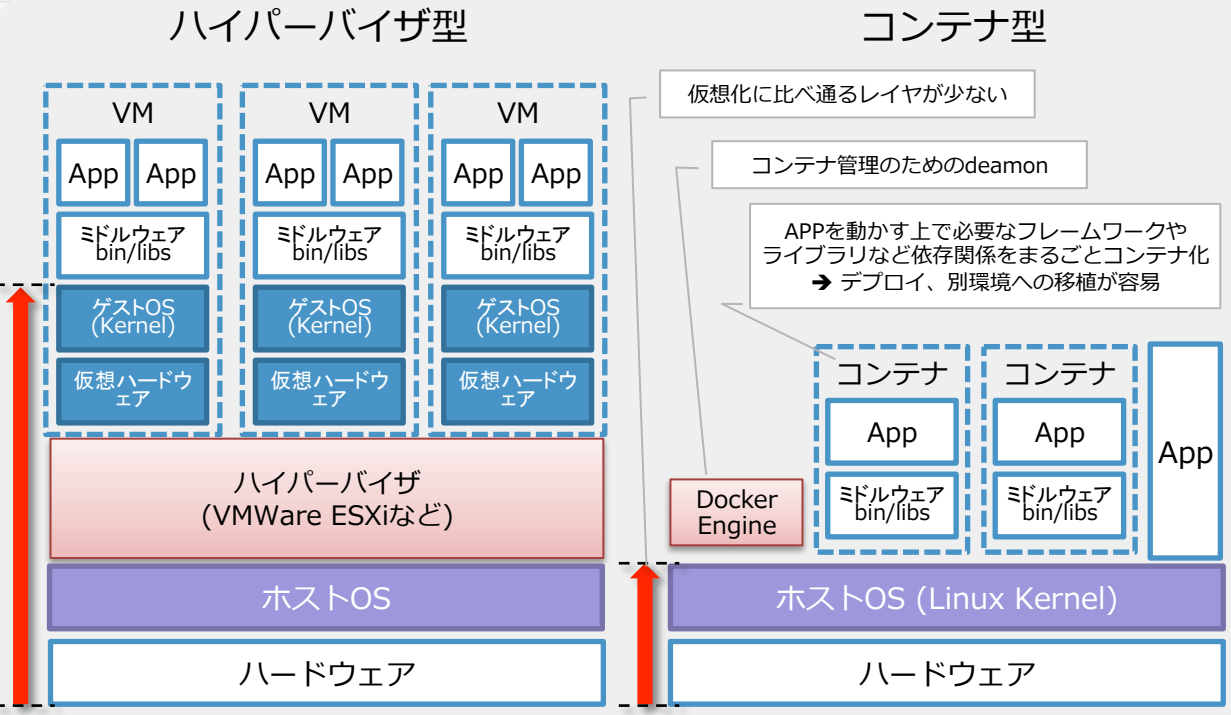
redhat.

マイクロサービスとコンテナ

- マイクロサービスアーキテクチャ原則
  - 自動化の文化 (DevOps)
  - 独立してデプロイ、スケール、リリースできる
  - アンチフラジャイル – プレッシャー下で堅牢性や回復力が向上
  - Polyglot (言語やフレームワークに依存しない)
  - 中央集権的でないデータ管理
- マイクロサービスの課題
  - 開発・構築
    - ビルド / デプロイの効率化、サービス間連携の問題
  - 運用
    - 障害対応、依存関係、リソース管理

マイクロサービスはコンテナで実現することで、
開発、運用管理の課題が軽減

従来のHW仮想化とコンテナ（Docker）の違い



■ 低負荷、高集約率、高速起動

- 低負荷、起動/停止が速い (ホストからはプロセスとして見えるだけ)
- ゲストOSやハードウェアエミュレートの負荷なし (ユーザ空間を隔離しているだけなので実装がシンプル)

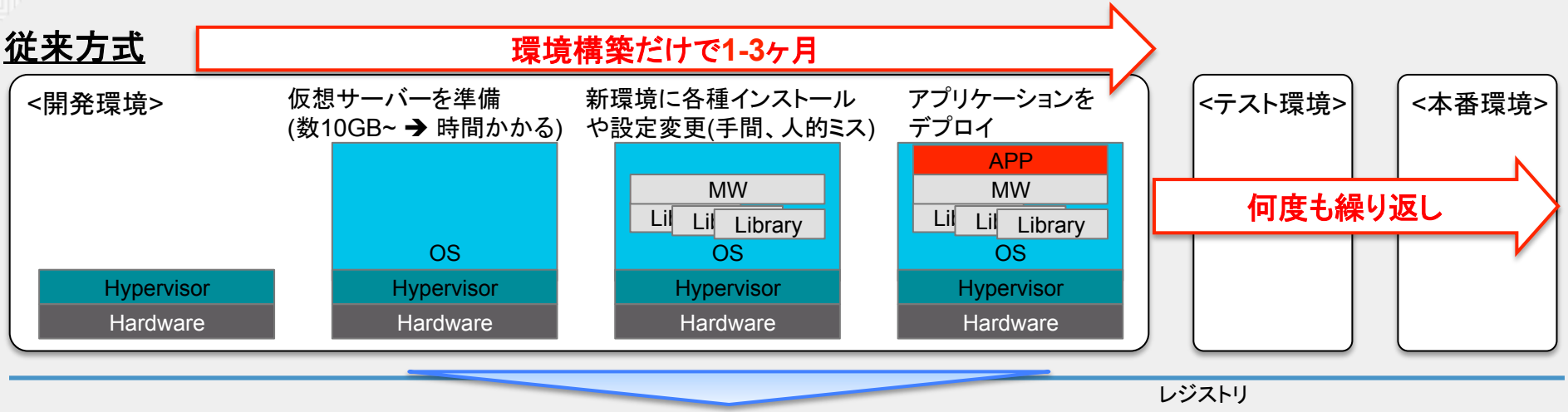
■ ポータビリティ

- Cloud(AWS, Azure, Google...)でもオンプレミスでもRHELが動くところならどこでも稼動 & 移植が容易
- 標準技術、ノーベンダーロックイン
- Git likeなバージョン管理機能とイメージ共有機構 (Docker Registry)
- アプリケーションの素早い展開と運用の両立(DevOps)

■ 新しい環境をすぐ用意

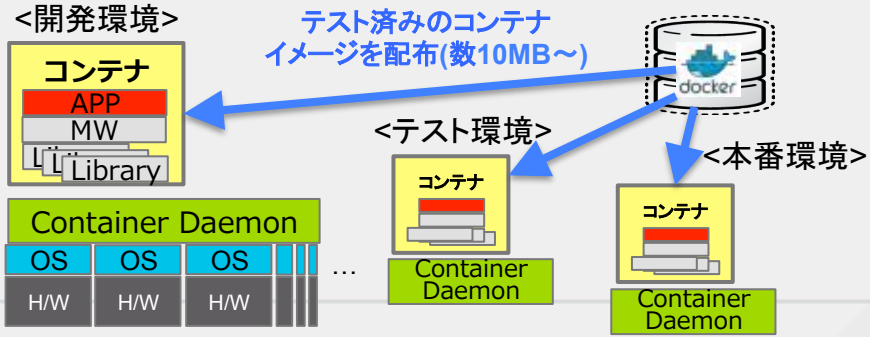
- 設定ファイルから何度も同じコンテナを作成可能 (Infra as Code)
- 稼働中のコンテナは修正しない。新しいコンテナと切替える(Immutable)
- インフラをコード化でき、構成管理を自動化しやすい

コンテナによるアプリケーションデプロイ方法



コンテナの場合

- 環境構築時間の大幅な削減
- 試行環境を何度でもいくつでも迅速に提供
- 柔軟なオートスケールの実現
- マルチ(ハイブリッド)クラウドの実現



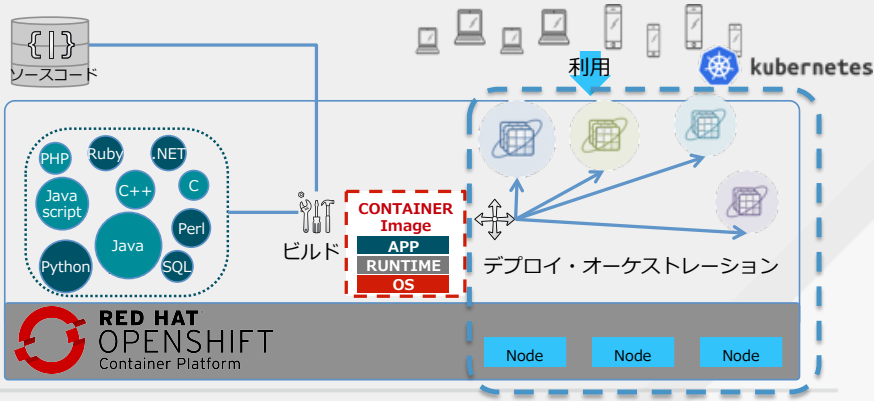
Red Hat OpenShift Container Platform

コンテナ化されたアプリケーションのためのビルド/実行/管理の基盤

RED HAT® OPENSHIFT Container Platform

- OpenShift Web UI
- 様々なPaaSサービス
  複数開発言語、DB, CI/CD,DevOps
- コンテナ・オーケストレーション
  Kubernetes
- コンテナ API
  Docker
- コンテナOS
  RHEL 7.3+

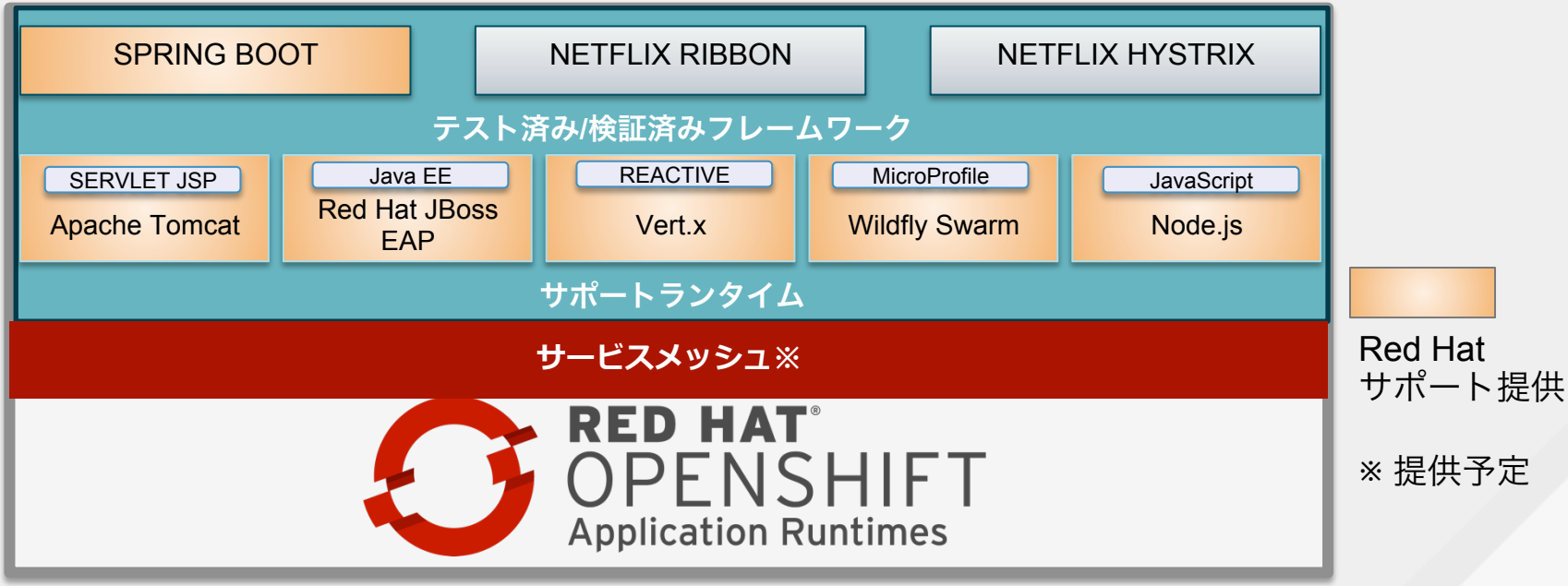
- ユーザエクスペリエンス
- 開発の効率化
- 環境の標準化
- 運用の利便性向上
- 安心の実行環境



redhat.

Red Hat OpenShift Application Runtimesとは

Red Hat OpenShift Application Runtimes(RHOAR)は、クラウドネイティブのサービスを構築・実行するマイクロサービスアーキテクチャーの実現のために必要なランタイムとアプリケーションフレームワークと実行基盤を提供。そして、サービスメッシュでマイクロサービスの運用に必要な仕組みを提供



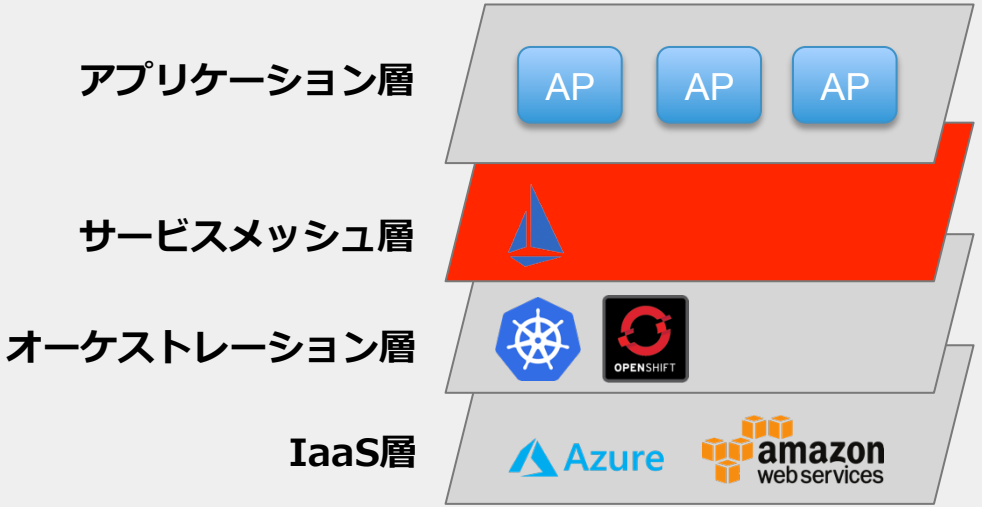
マイクロサービス課題への対応: サービスメッシュ

- マイクロサービス間の連携を透過的に扱うためのレイヤーを提供
  - HTTPやgRPCなどの低レイヤーの通信を隠蔽(タイムアウト、リトライ等)
  - サービスディスカバリとルーティング、トラフィック制御
  - サービス障害の影響を最小化 (例: circuit breaker)
  - サービス間認証/認可、暗号化
  - サービス全体のモニタリングとレポーティング

- Kubernetes(コンテナ基盤)上の新たな「ミドルウェア」
  - Istio: https://istio.io/
  - linkerd: https://linkerd.io/
  - Conduit: https://conduit.io/

「サービスメッシュ層」という概念

- サービスの発見、ルーティング
- サービスの認証
- アクセス数制限
- ロードバランス
- リトライ、サーキットブレイク
- Blue/Green、カナリアリリース
- リクエストのトレース
- サービスのメトリクス
- エラー時の挙動確認



サービス間通信とPod/Containerを切り離す
＝ サービスメッシュ層で設定可能にする

アプリケーション開発とデリバリーをスピードアップ
～RHOAR の効果～

課題	提供内容	効果
分散アプリケーション開発の困難 (アーキテクチャ)	• OpenShift とMiddleware servicesとの統合 • アプリケーション サービス公開 • アプリケーション要件を満たすための機能や補完機能	**単純な開発**
1 つのアプリの要件に引きずられるリスク (プラットフォーム)	• マルチクラウドでのデプロイ可能 • マルチランタイムとマルチフレームワーク • 軽量アプリケーション	**戦略的な柔軟性**
サイロな開発、テスト、デベロップメントとウォーターフォールプロセス (プロセス)	• CI/CD ツールとの統合　（Git, Jenkins , Maven など） • DevOpsパイプラインの自動化（A/B, カナリアリリースなど） • 開発ツールのOpenShift.io 連携	**DevOps 自動化**
マイクロサービス運用の複雑化	• サービス認証 • サービス呼び出しのロードバランス • サービス呼び出しエラーの対応	**運用の自動化**

OpenShift Roadmap

OpenShift Container Platform 3.10 (July)
- Kubernetes 1.10 and CRI-O option
- Smart Pruning
- **Istio (Dev Preview)** — Dev Preview
- oc client for developers
- Golden Image Tooling and TLS bootstrapping
- Windows Server Containers (Dev Preview))
- Prometheus Metrics and Alerts (Tech Preview)
- S3 Svc Broker

OpenShift Online & Dedicated
- Dedicated self-service: RBAC, limit ranges
- Dedicated encrypted storage, multi-AZ, Azure beta

OpenShift Container Platform 3.12 (Dec/Jan)
- Kubernetes 1.12 and CRI-O default
- Converged Platform
- Full Stack Automated Installer
  - AWS, RHEL, Azure, OSP
- Over the Air Updates
- RHCC integrated experience
- Windows Containers GA
- Easy/Trackable Evaluations
- Red Hat CoreOS Container Linux with Ignition Automations
- Cluster Registry
- HPA metrics from Prometheus

OpenShift Online & Dedicated
- Cluster Operator driven installs
- Self-Service Dedicated User Experience

Q2 CY2018 | **Q3 CY2018** | **Q4 CY2018** | **Q1 CY2019**

OpenShift Container Platform 3.11 (Sept)
- Kubernetes 1.11 and CRI-O default
- Infra monitoring, alerting with SRE intelligence, Node Problem Detector
- Etcd, Prometheus, and Vault Operators - Tech preview
- Operator Certification Program and JBoss Fuse Operator
- Autoscaler for AWS and P-SAP features
- Metering and Chargeback (Tech Preview)
- HPA Custom Metric
- Tech preview of ALM
- New web console for developers and cluster admins
- Ansible Galaxy ASB support
- CNV (Tech Preview)
- OVN (Tech Preview for Windows)
- FIPS and other Security PAGs

OpenShift Online & Dedicated
- OpenShift Online automated updates for OS
- Chargeback (usage tracking) for OpenShift Online Starter

OpenShift Container Platform 3.13 (March)
- Kubernetes 1.13 and CRI-O default
- Full Stack Automation
  - GCP, VMware
- **Istio GA** — GA
- Mobile 5.x
- Serverless (Tech Preview)
- RHCC for non-container content
- Integrated Quay (Tech Preview)
- Idling Controller
- Federated Ingress and Workload Policy
- OVN GA
- Che (Tech Preview)

OpenShift Online & Dedicated
- OpenShift.io on Dedicated (Tech Preview)

マイクロサービスの課題はほぼ解決

- 構築
  - サービスの境界 / 粒度
  - ビルド / デプロイの効率化
  - サービス間連携 (サービスディスカバリ、オーケストレーション、コレオグラフィー)
  - テスト (サービス単位 / エンドツーエンド)
  - 自律的な回復
- 運用
  - サービスの健全性の管理 / 監視
  - サービス障害対応 (ログ集中管理、トレーサビリティ)
  - サービス間の依存関係の管理 (バージョニング、データマイグレーション)
  - パフォーマンス管理 / リソース管理
  - API管理 (特に外部公開用)
- セキュリティ
  - 分散環境での認証 / 認可
  - 暗号化
  - 監査

Red Hat OpenShift Container Platformで提供
Red Hat OpenShift Container Platformで提供予定
Red Hat 3scale API management Platform で提供

マイクロサービスアーキテクチャを導入して成功した事例

ベライゾン事例（Lift & Shift)

ITモダナイゼーションの課題

verizon

ビジネスを取りまく環境の変化にいつでも追随できる
ことができるようなITプラットホームの実現が必至

1. スケールアウト、機能拡張させるにコストがかかるベンダーのミドルウェアを利用していた
2. サーバの仮想化は導入するもデマンドに応じてコンピュータリソースを伸縮自在させることができていないアプリケーション
3. サービスリリースのための本番環境の切替えに時間がかかっている
4. 開発の効率化のためCI/CDの導入を検討する必要
5. ベンダーロックインのリスク回避
6. パブリッククラウド有効活用（リソースバースト時）ができていない（パブリック移行に課題）

redhat.

ITモダナイゼーションの解決策

verizon

マイクロサービスの導入

- ビジネスニーズの変化に追随するアプリケーション（サービスの再利用性を促進することで、不要な新規開発を削減）
- **すべてのレガーシアプリケーションをコンテナー化し、マイクロサービスベースのアプリケーションにする**
- **オープンソースベースのミドルウェア製品(JBoss,Postgress)へのマイグレーション（オープンソース採用でベンダーロックインを回避）**

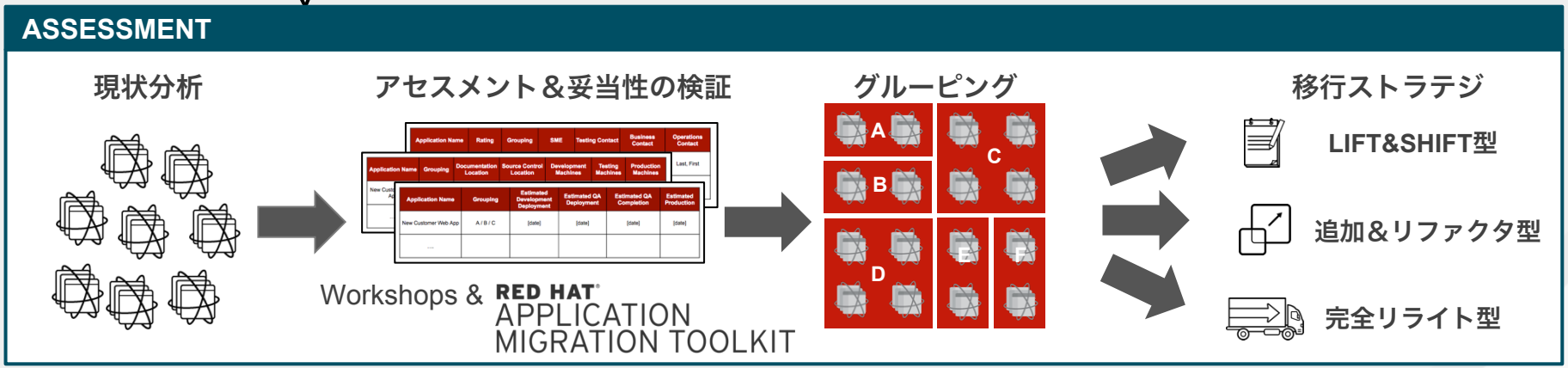
イミュータブル・インフラストラクチャー

- 本番サーバを使い捨て可能な環境→すべてのアプリケーションのコンテナー化
- 「Blue-Green Deployment」のリリース手法導入

マルチクラウドへの対応

- プライベートーパブリック間のポータビリティー性

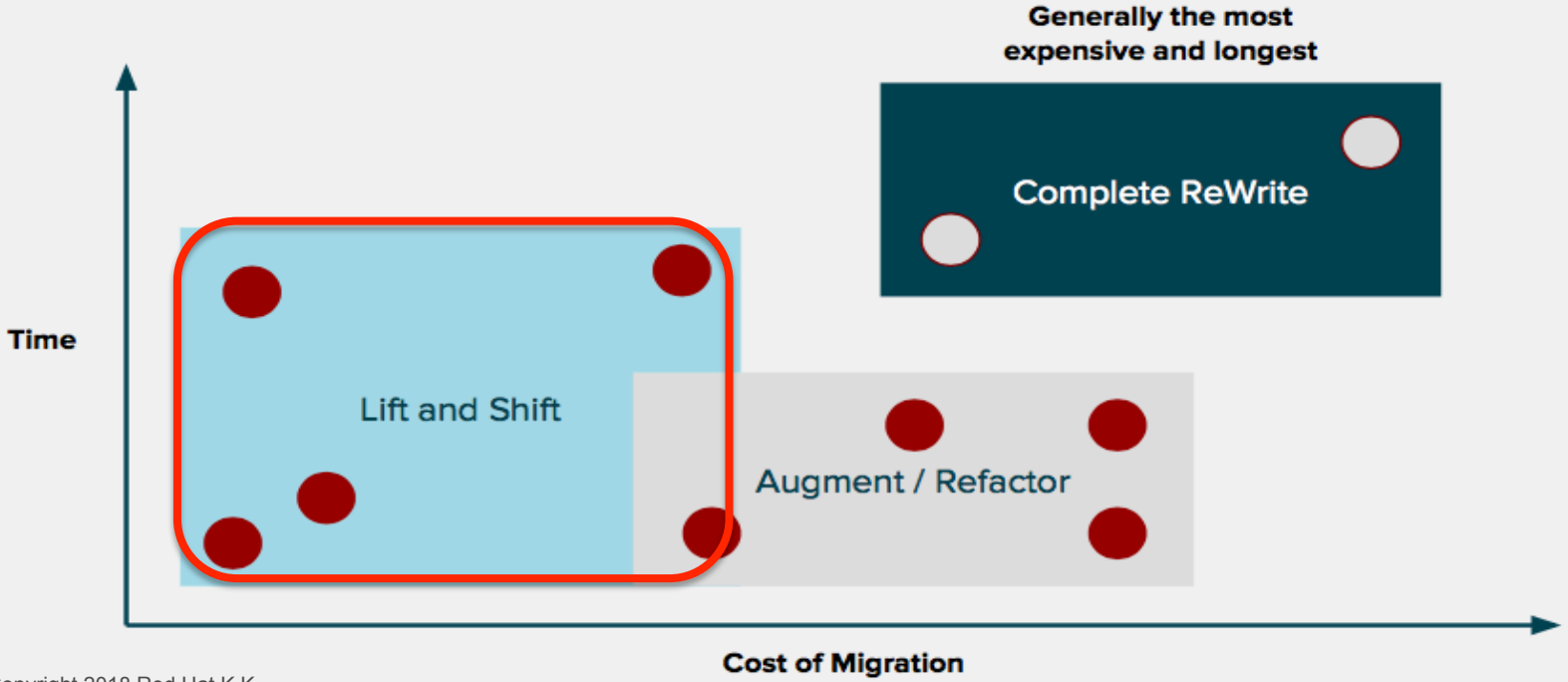
既存システムをクラウド移行するレッドハットの手法



マイクロサービス移行への３つの方法

verizon

1. **リフト・アンド・シフト**　基本ソースコードの変更は行わないでコンテナ化
2. **機能追加とリファクタリング**　基本機能はそのままで連携のためのコードを追加してコンテナ化
3. **完全リライト**　コンテナを最大限に利用するようにソースコードを完全に変更



redhat.

OpenShift 導入効果

verizon

1. アプリケーションがスケールアウトできるアーキテクチャとなり、オートスケール化を実現
2. 効率的なリリース管理（ダウンタイムなしのリリース、瞬時のバージョン切替え）
3. 開発生産性の向上（自動化されたCI/CD）
4. バースト時のパブリッククラウドの利用
5. ベンダーロックインのリスク排除

redhat.

レッドハットのデジタル変革支援ソリューション

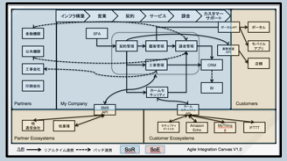
redhat.

レッドハットのデジタル変革支援ソリューションメニュー

アジャイルAPI インテグレーション

ビジネスバリューチェーンとSoE/SoRの関係性を明確にし、マイクロサービス化による最適なSoE基盤への移行パスを提案します。

Agile Integration Canvas

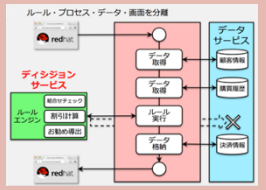


API Model Canvas

Miles & More
ルフトハンザ航空からカーブアウトしたマイレージサービス提供会社で、API中心のSoE基盤を構築しパートナーシップを300社以上に拡大

ビジネス モダナイゼーション

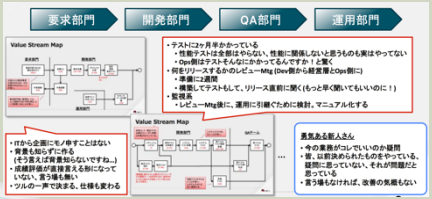
ビジネスの変化に追従できるビジネスプロセスを実現するため、システムをプロセス・ルール・データに分離する、最適なアーキテクチャーを提案します。



iSmartテクノロジーズ
トヨタの部品製造会社がIoTで設備投資2億円を削減
IoTサービス事業会社を設立して、改善コンサルティングからIoTデータ分析サービスまで提供

DevOps

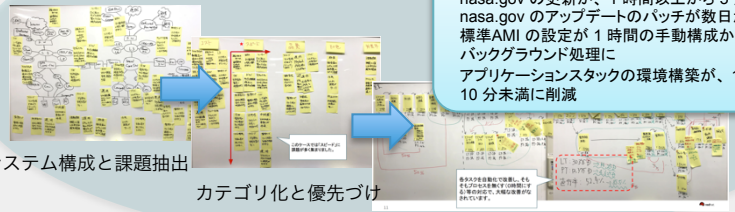
アプリ開発と運用における課題点を洗い出し、バリューストリームマッピングにより改善点を明確にしながら、お客様の組織・プロセス・文化の改革のロードマップを提案します。



SoftBank
DevOpsの手法の活用（DevOps Discovery Workshopを活用）
コンテナを活用したCI/CDパイプラインを構築、2週間に1回のデプロイ回数が、118回と大幅に増加。
テストカバー率が20%から90%に向上し品質も向上

IT オートメーション

運用管理のプロセスにおける課題点を洗い出し、組織全体で最適な自動化の手段と手法を提案します。

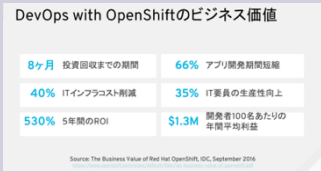


システム構成と課題抽出　カテゴリ化と優先づけ　バリューストリームマッピングによる改善効果の定量化

NASA
nasa.gov の更新が、1 時間以上から 5 分未満へ
nasa.gov のアップデートのバッチが数日から 45 分へ
標準AMI の設定が 1 時間の手動構成からシームレスなバックグラウンド処理に
アプリケーションスタックの環境構築が、1 ～ 2 時間から 10 分未満に削減

ハイブリッド クラウド

アプリケーションのワークロードを最適な環境に自在に配置でき、インフラコストの全体最適化を実現するアーキテクチャーを提案します。



ドイツ銀行
ベンダーフリーのオープンな環境（ハイブリッドクラウド）でEverything as a Service(ハード、ソフト、パートナー全てをサービスにする)を実現する基盤を採用
2020年に約1000億円のコスト削減を目指す

まとめ

まとめ

なぜ今マイクロサービス

- 技術が成熟し、マイクロサービスが考え方だけでは無く、実際の運用まで実現できるようになってきている

Red Hat OpenShift Container Platform による実現

- マイクロサービスのシステムを構築・運用するにはコンテナ基盤が重要
- Red Hat OpenShift Container Platformはエンタープライズで利用できるレベルでのコンテナ基盤を提供

レッドハットのデジタルトランスフォーメーション支援

- レッドハットの豊富な経験と深い知識を持ったコンサルタントがお客様のDXの実現を強力にサポートします

ご静聴ありがとうございましました

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos

redhat.