



Azure DevOps (VSTS) 概要

日本マイクロソフト株式会社

クラウドソリューションアーキテクト

赤間 信幸

<https://blogs.msdn.microsoft.com/nakama/> @nakama00

Microsoft Corporation



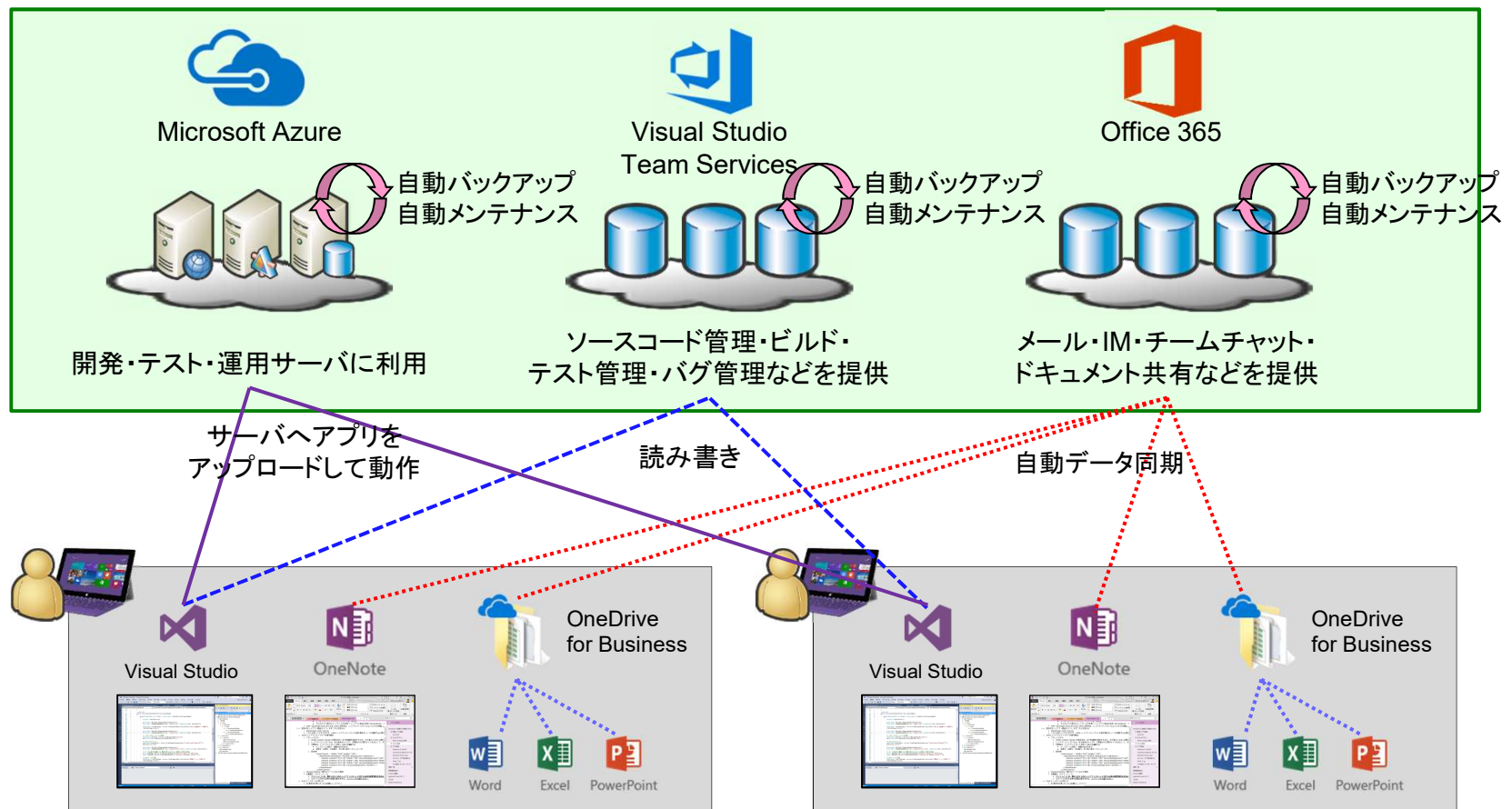
Azure DevOps (VSTS) 概要

- VSTS (Visual Studio Team Services) がどのようなものであるのかを理解する
 - VSTS 概要
 - 昨今の SI 開発現場によくある課題と解決の方向性
 - 近代的開発・テスト環境 リファレンス構成

VSTS (Visual Studio Team Services) とは

■ システム開発における資産管理とタスク管理を行うためのクラウドサービス





















































- Azure : 開発・テスト・運用サーバとして利用
- VSTS : ソースコード管理・ビルド・テスト管理・バグ管理・タスク管理などに利用
- Office 365 : コミュニケーション & コラボレーション、ドキュメント作成・管理などに利用



VSTS (Visual Studio Team Services) とは

■ 多様な選択肢と柔軟性の提供

- Azure や VSTS は、多彩な開発言語・ツール・ミドルウェアとの連携が可能
- 今お使いのツールと言語そのままクラウドのパワーを！

DevOps										
Management										
Applications										
App frameworks & tools										
Databases & middleware										
Infrastructure										

昨今の SI 開発現場によくある課題と解決の方向性

- 日本の SI 開発現場でよくみられる課題の中には、近代的 SI 技法によりすでに解決されているものも数多く存在する
 - 代表的な課題として、以下の 4 つを取り上げる

よくある課題

① 実装状況のブラックボックス化

- ・ 納品直前 or 後に品質の悪さが発覚
- ・ ニアショアや分散拠点開発にて、開発状況が追跡できない(ブラックボックス化)

② 長期保守の困難性

- ・ 保守作業の引き継ぎや長期保守が大変
- ・ バグ修正・仕様変更に時間がかかる & ベンダー見積もりが非常に高い

③ 大部屋開発の生産性の低さ

- ・ 増員や減員にスムーズに対応できない
- ・ 拠点集約しているにもかかわらず、コラボレーション効果が出ていない

④ プロパー社員の高労働負荷・低生産性

- ・ 場所・時間に縛られた労働が多く、社員の多様なワークスタイルに対応できない
- ・ SI 手法が古く、生産性を高められない

近代的 SI 手法による解決

① クラウド活用による開発状況の可視化

- ・ クラウド上での資産・タスク情報の管理
- ・ カナリアサーバ利用による開発状況の可視化と容易な把握

② 納品物の再定義による長期保守の実現

- ・ ビルド・テスト・リリースの自動化・標準化
- ・ 開発 PC の DaaS 化による環境凍結
- ・ 納品物標準化による容易な保守の実現

③ 標準化されたツールとアジャイル活用

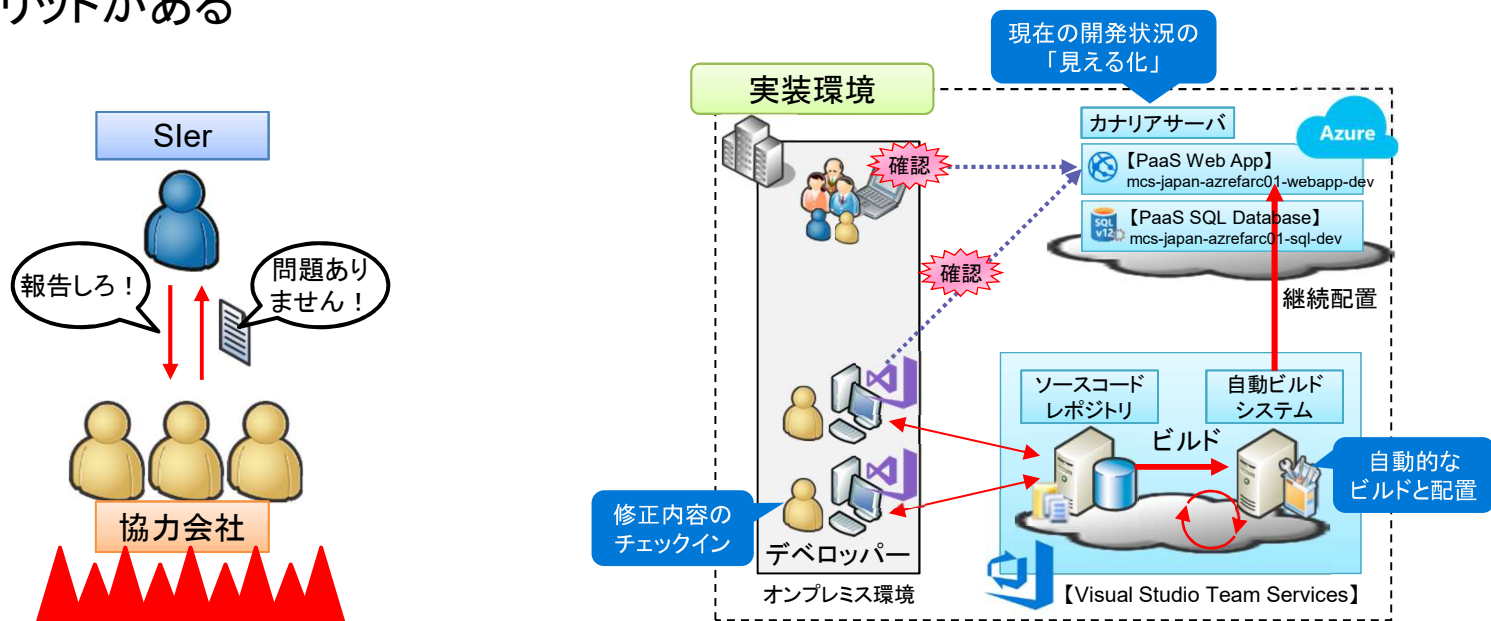
- ・ 標準化されたツールチェーンがインストールされた開発 PC による開発作業
- ・ アジャイルによるチームワークの活性化

④ フレキシブルワークスタイルの導入

- ・ 場所・時間に縛られない情報・知識共有
- ・ 開発基盤のクラウド化によるリモートからのデータ参照の実現

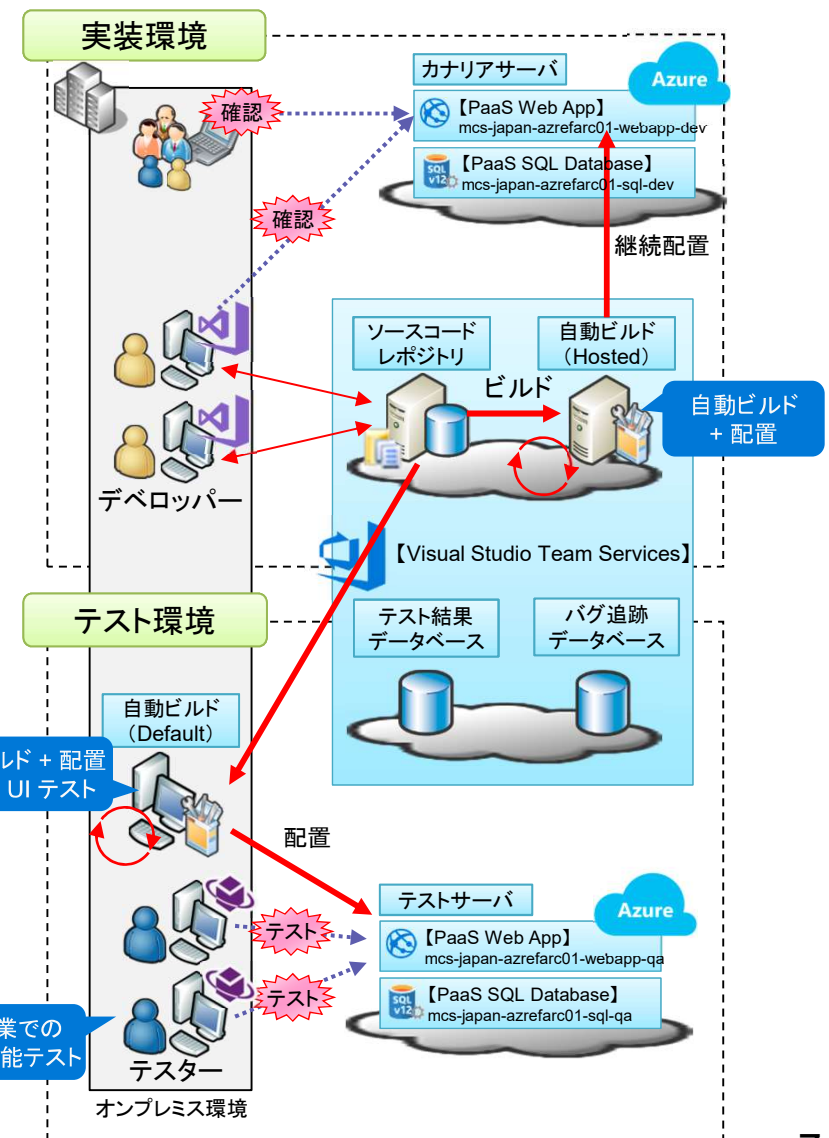
① クラウド活用による開発状況の可視化

- クラウドにコード資産やタスク情報を保存し、さらに CI/CD (カナリアサーバ) で実物を確認できるようにすることで、開発状況のブラックボックス化を防ぐ
 - コード資産やタスク情報を VSTS レポジトリに保存する
 - これにより、リモートでも最新のコードの状況やタスクの状況の確認ができるようになる
 - CI/CD (カナリアサーバ) により『動く実物』をいつでも見られる状況にする
 - ソースコードやタスク情報だけから、開発状況や品質を推測することは難しい
 - 『動く現物』は、少し触ってみるだけで、だいたいの品質が直感的に把握できる
- 開発状況の可視化に加え、ニアショア開発などへの対応がしやすくなる点で大きなメリットがある



② 納品物の再定義による長期保守の実現

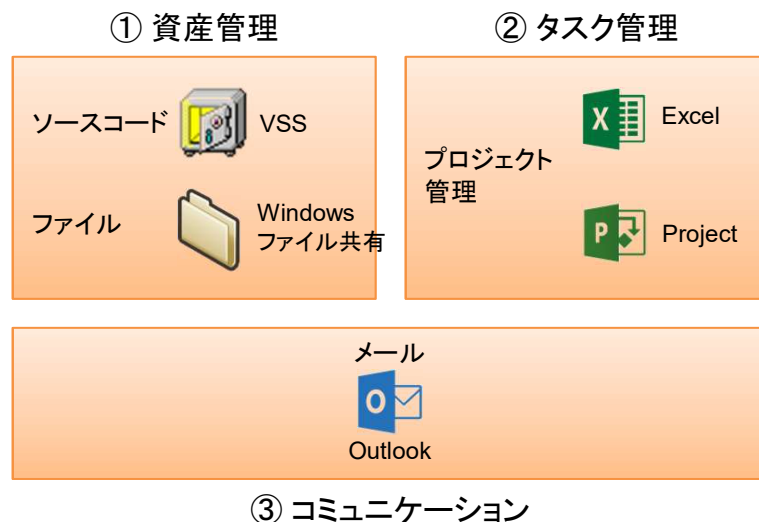
- 現在のベンダーからの納品物やその形態は、長期保守を意識していない
 - システムごとに納品形態が異なるため、保守フェーズのやり方がバラバラになる
 - ビルド・テスト・リリースが自動化されていないため、簡単な修正でも非常に大変
- この問題の解決のため、VSTS を活用し、ビルド・テスト・リリースを標準化する
 - 個別最適化することなく、標準的な基盤の上に作った自動ビルド・自動テスト・自動リリースの仕組みを納品してもらう
 - これにより、コード修正時の再リリースコストを大幅に低減する
- さらに、コーディング・テスト用 PC を Azure IaaS によって DaaS 化してしまう
 - コーディング用 PC やテスト用 PC を DaaS 化することにより、環境凍結が容易になる
 - これにより、長期保守の容易化を図る



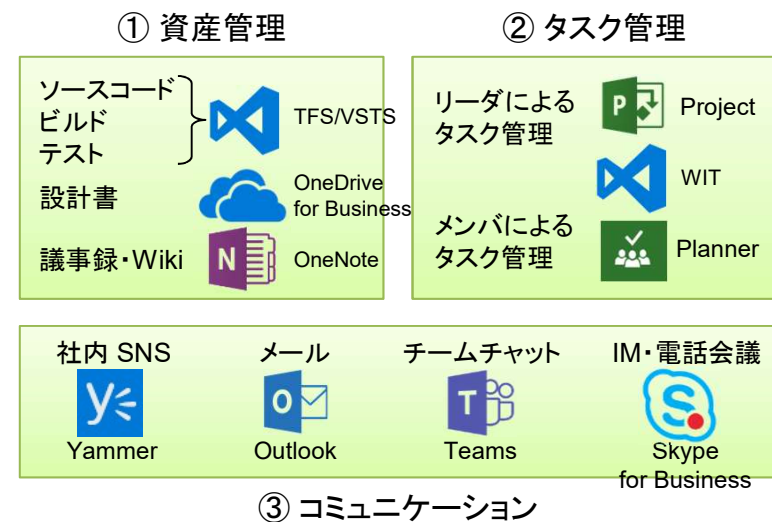
③ 標準化されたツールとアジャイル活用

- 現在の大部屋開発では、生産性を削ぎかねない煩雑な作業標準が横行している
 - 過度に制限された環境（インターネットでの調べものできない等）
 - 増員・減員にスムーズに対応できないレガシーな環境
 - ファイルサーバ + Excel 台帳での非効率的な資産管理、etc...
- 標準化された近代的ツールセットとアジャイルプラクティスの活用により、作業担当者のやる気と生産性を引き出す
 - 標準化されたツールチェーンがプリインストールされた開発 PC の用意
 - 近代的なコミュニケーション・コラボレーションツールの提供
 - アジャイルプラクティスの活用によるチームワークの活性化

Before : 昔の開発環境



Current : 現在の開発環境



④ フレキシブルワークスタイルの導入

- 優秀なプロパー社員を失わないための、魅力的な職場環境の整備が必要
 - 現在の日本の SI 開発現場では、労働集約的な労務管理スタイルが横行している
 - このため、場所・時間に縛られた労働が極めて多い
 - ライフスタイルに併せたワークスタイルの調整ができず、特に育児や介護などの在宅勤務ニーズなどに対応していくことが全くできていない
 - クラウド環境とモバイルデバイスの活用によりフレキシブルワークスタイルを実現することで、優秀なプロパー社員に対して魅力的な職場環境を提供し続ける
 - 場所・時間に縛られない、データやナレッジの共有・やり取り
 - どこからでもデータ参照を可能とすることで、素早い一次対応を実現



近代的な開発環境に向けたロードマップについて

アプローチ フェーズ	Step 1 ベンダーの 管理性向上	Step 2 保守の容易化	Step 3 大部屋開発の 生産性改善	Step 4 プロパーのフレキシ ブルワーク対応
主な狙い	<ul style="list-style-type: none"> ニアショア開発対応 リアルタイム PJ 状 況・品質トラッキング 	<ul style="list-style-type: none"> ベンダーからの保守 引取の容易化 長期保守の容易化 	<ul style="list-style-type: none"> 柔軟な要員増減 社員・協力会社の活 性化と生産性改善 	<ul style="list-style-type: none"> 変則的就業対応 リモート管理・保守 社員の離職防止
主にやること	<ul style="list-style-type: none"> VSTS 上での資産・ タスクの管理 カナリアサーバ導入 	<ul style="list-style-type: none"> 開発 PC の仮想化 ビルド・テスト・リリー ス自動化 	<ul style="list-style-type: none"> O365 インフラ導入 アジャイルプラクティ スの一部活用 	<ul style="list-style-type: none"> デバイスの最新化 リモートからの環境 参照・操作
導入する主な クラウドツール	VSTS ソースコード管理、タスク管理 VSTS ビルド・テスト・リリース自動化 Azure PaaS Web Apps, SQL データベース Azure DaaS (開発端末の仮想化) O365 Teams, Skype, OneNote			
ツール導入 以外の 主な施策	<ul style="list-style-type: none"> ワークフロー定義 ネット回線整備 	<ul style="list-style-type: none"> 標準納品物の定義 SI ナレッジ最新化 	<ul style="list-style-type: none"> メンタリティ・モラル 改善 社内規定の最小化 	<ul style="list-style-type: none"> 就業規則の改善 成果ベースの評価 ロールキャリア整備
適用範囲・対象 プロジェクト	先行パイロット PJ (小規模、1~2 個)	横展開 (小中規模、10 個)	大規模 PJ 適用 (大規模、1~2 個)	全社展開
ベースライン 育成施策	アーキテクトの増員・増強 SE・プロマネの SI スキル・ナレッジの最新化 部課長クラスの意識改革			

近代的開発・テスト環境 リファレンス構成について

- 開発・テスト環境を考える際は、以下の 2 側面から検討・整理する必要がある
 - ① 成果物ワークフロー（論理構成図）
 - 誰がどのような作業を行い、どのような成果物を誰に引き継いでいくのか？
 - 成果物ワークフローに関するベストプラクティスは、どのような開発形態であってもほぼ同じになる
 - ② 開発・テスト環境構成（物理構成図）
 - ①のワークフローを、どのようなサーバ構成やクラウドサービスによって実現するか？
 - 物理配置構成は、セキュリティポリシーや開発体制によって大きく変わる
- このため、上記 2 つのイラストを別のイラストとして作成することが必要となる
 - ① → いずれの場合も基本的に同じ
 - ② → 開発要件により異なるため、代表例として 4 パターン示す
 - パターン A. 協力会社の引き込み開発
 - パターン B. 協力会社の引き込み開発（構成テストつき）
 - パターン C. オフショア・ニアショア開発（構成テストつき）
 - パターン D. 大部屋開発（構成テストつき）

近代的開発・テスト環境 リファレンス構成について

■ ① 成果物ワークフロー(論理構成図)



Azure のサービスを利用



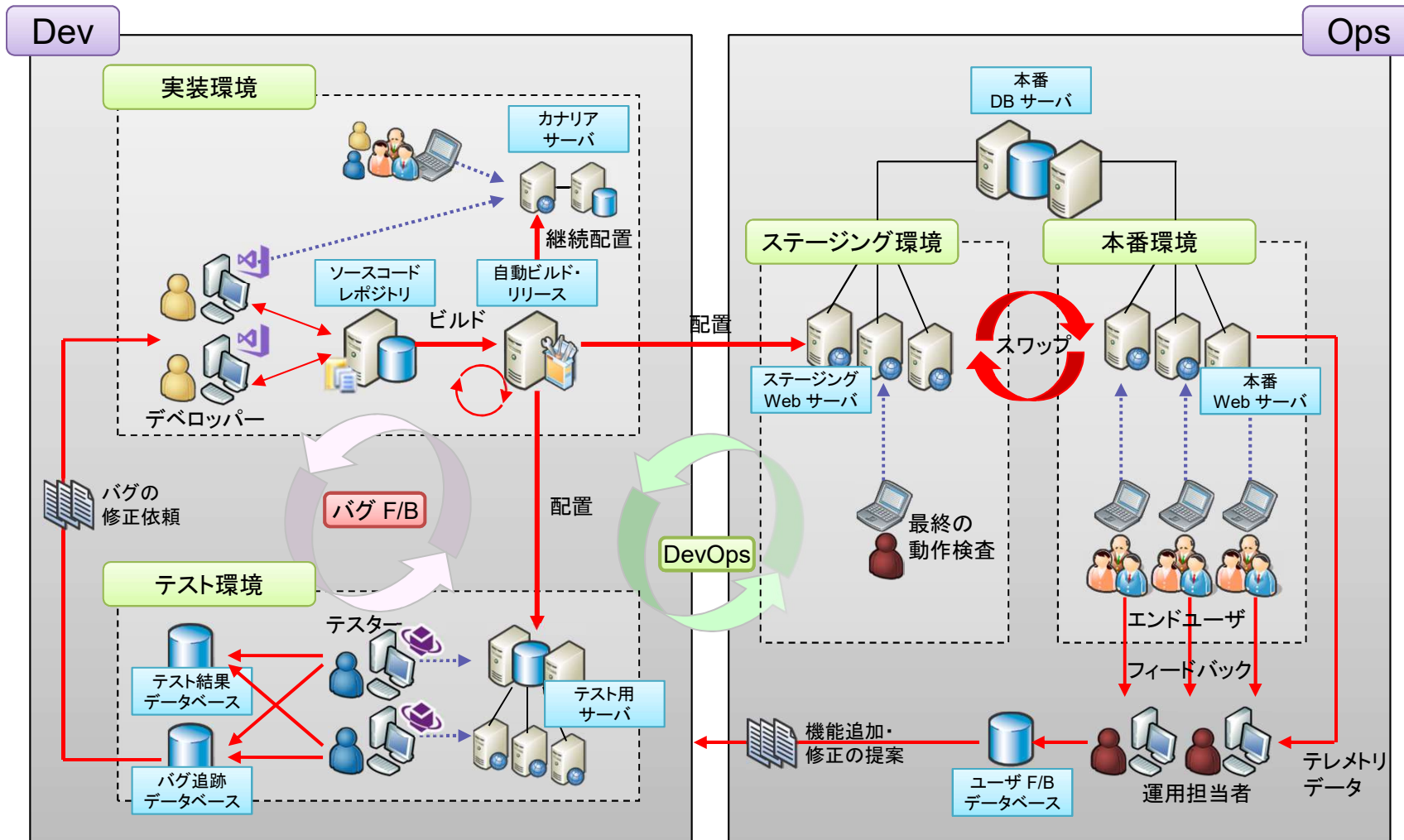
VSTS のサービスを利用



Visual Studio



Test Manager



近代的開発・テスト環境 リファレンス構成について

■ ② 開発・テスト環境構成(物理構成図)

□ パターン A. 協力会社の引き込み開発



Azure のサービスを利用



VSTS のサービスを利用



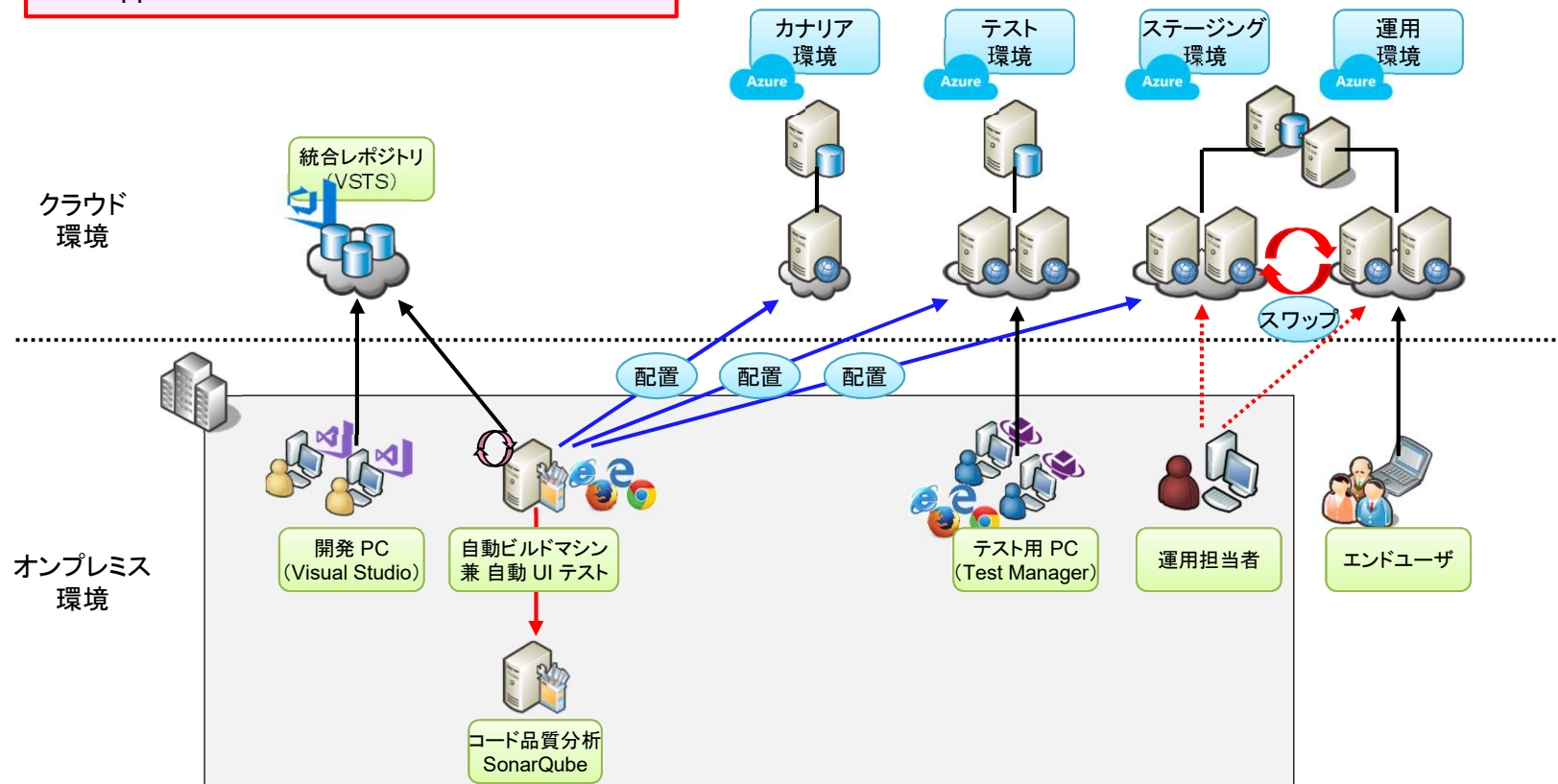
Visual Studio



Test Manager

[構成上のポイント]

- ビルドマシンやコード品質分析マシンをオンプレ側に置いてしまう(セキュリティ確保が容易)
- カナリア・テスト環境については Azure Web Apps や SQL DB を利用する(構築がラク)



近代的開発・テスト環境 リファレンス構成について

■ ② 開発・テスト環境構成(物理構成図)

□ パターン B. 協力会社の引き込み開発 (構成テストつき)



Azure のサービスを利用



VSTS のサービスを利用



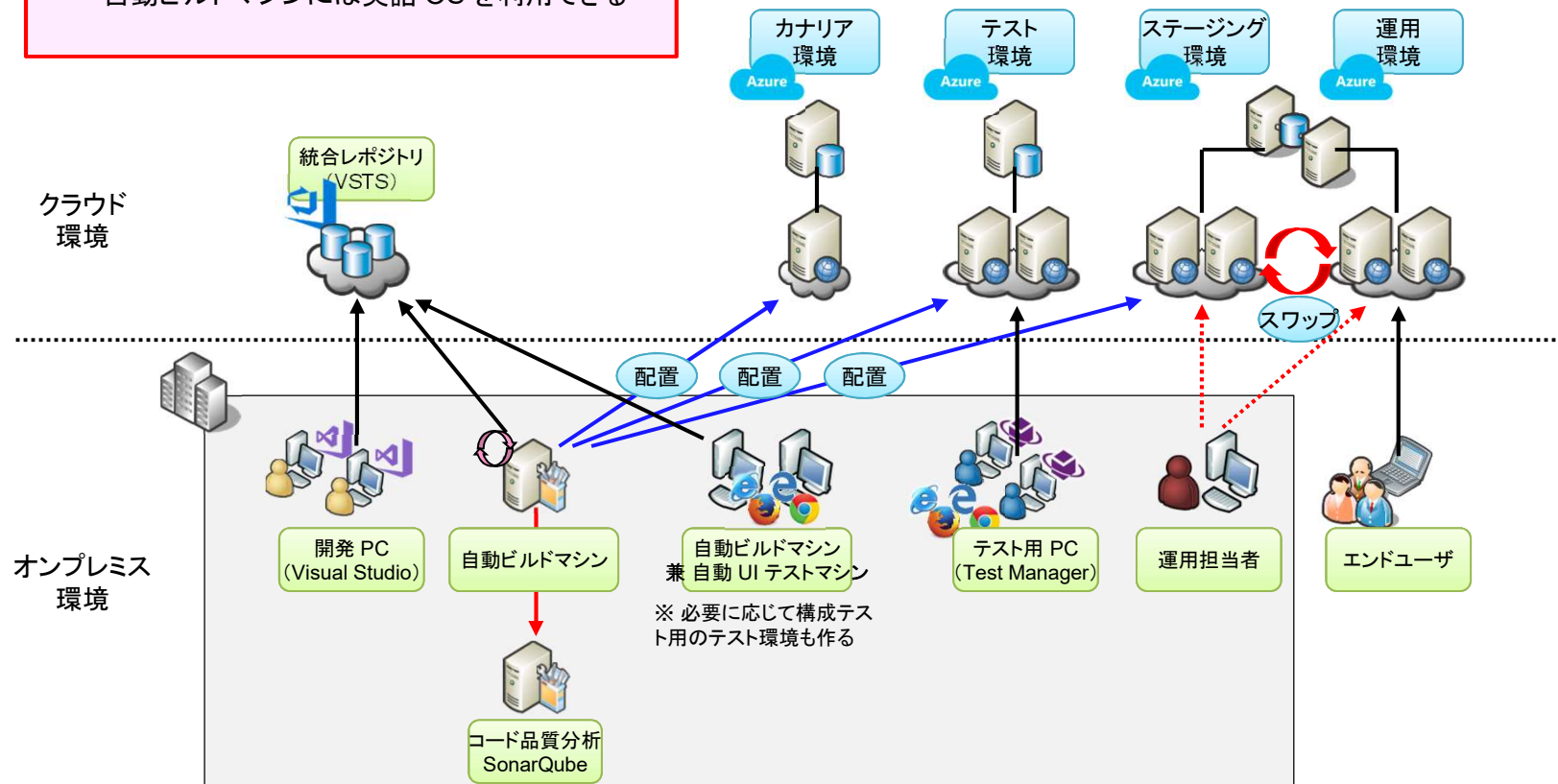
Visual Studio



Test Manager

[構成上のポイント]

- 自動 UI テストマシンをビルドマシンから分離し、Win7, 10 など複数の OS で構成する
- 自動ビルドマシンには英語 OS を利用できる



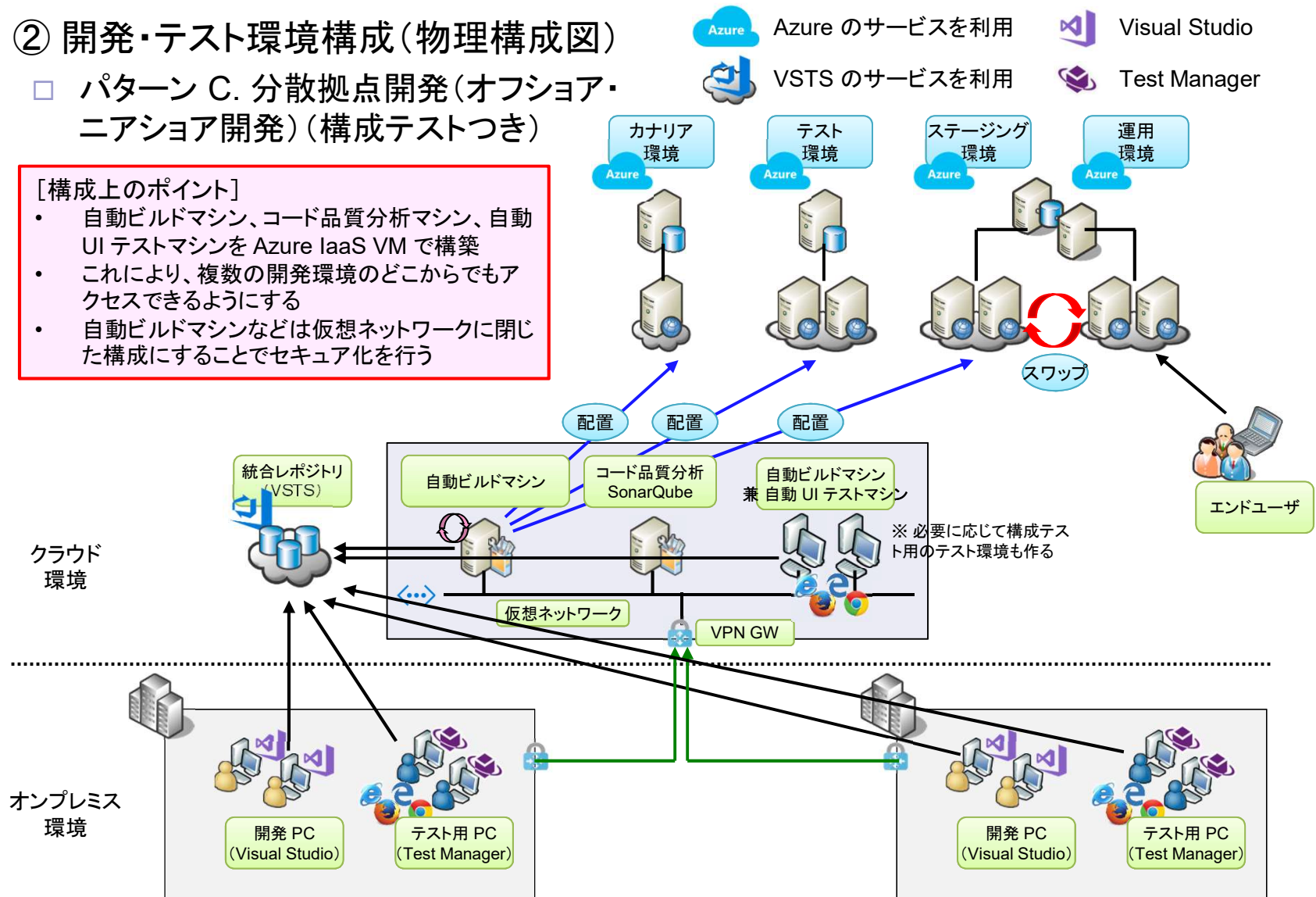
近代的開発・テスト環境 リファレンス構成について

② 開発・テスト環境構成(物理構成図)

□ パターン C. 分散拠点開発(オフショア・ニアショア開発)(構成テストつき)

[構成上のポイント]

- 自動ビルドマシン、コード品質分析マシン、自動 UI テストマシンを Azure IaaS VM で構築
- これにより、複数の開発環境のどこからでもアクセスできるようにする
- 自動ビルドマシンなどは仮想ネットワークに閉じた構成にすることでセキュア化を行う



近代的開発・テスト環境 リファレンス構成について

■ ② 開発・テスト環境構成 (物理構成図)

□ パターン D. 大部屋開発 (構成テストつき)



Azure のサービスを利用



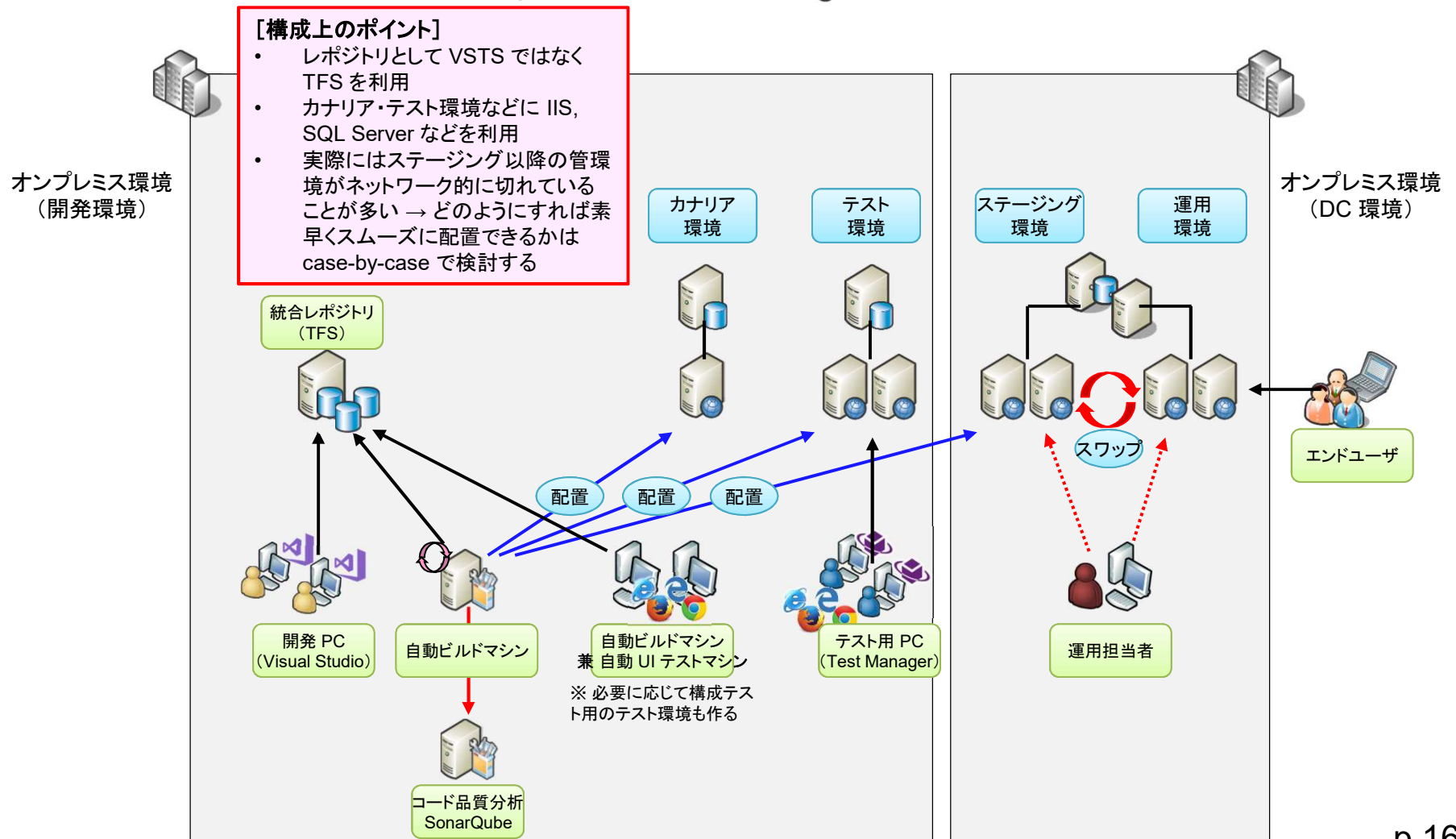
Visual Studio



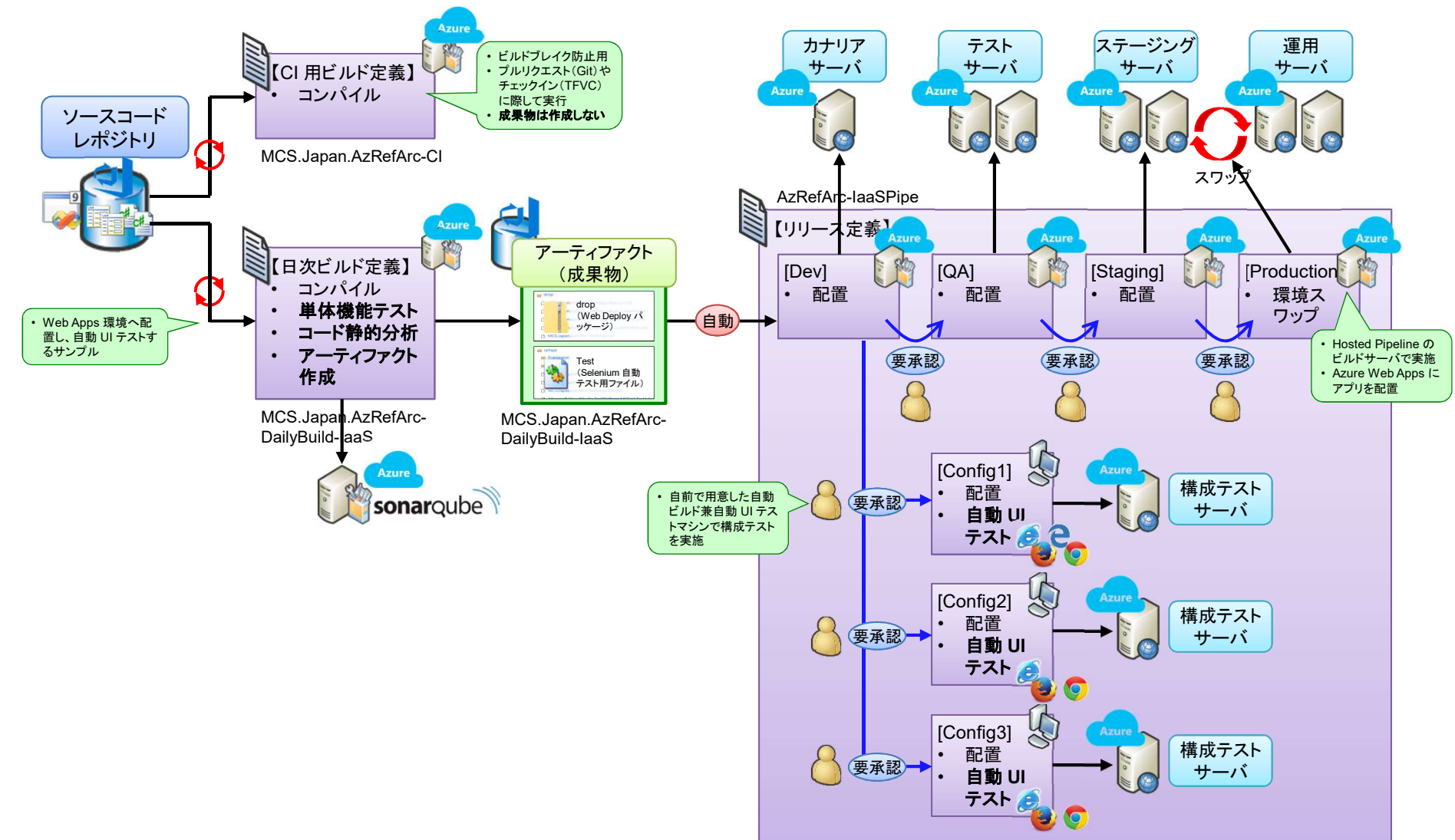
VSTS のサービスを利用



Test Manager



Response	Percentage
Yes, the current government is responsible	92%
No, the current government is not responsible	8%





利用条件:

- 本書に関するすべての権利は、Microsoft Corporationおよびその関連会社(以下、マイクロソフト)が保有しています。本書は情報提供のみを目的としており、本資料に記載されている情報は、本資料作成時点での情報となります。
- 状況等の変化により、内容は変更される場合があります。マイクロソフトによる事前の承諾がない限り、本書の全部または一部を複製、改変、翻案、再頒布、公衆送信、貸与、譲渡したり、第三者に開示または共有することは、認められません。
- マイクロソフトは、本書の内容について何ら保証するものではなく、本書の使用に関連してお客様、お客様の関連会社、または第三者に生ずる間接的、付随的、結果的な損害(営業機会や営業情報の損失などを含む)について一切責任を負いません。
- 本書の中で例として使用されている企業、名前およびデータは、特に記述がない限り、架空のものです。

MICROSOFT CONFIDENTIAL

© 2018 Microsoft Corporation. All rights reserved.