67

ハイブリッドクラウド環境での社内開発環境への導入事例[1]

1. 概要

パブリッククラウドが広く普及したが、従来のオンプレミスやプライベートクラウドの既存資産を全てパブリッククラウドに移行するということはコスト面や運用面においても現実的ではないことが多い。本事例は、開発環境へハイブリッドクラウド環境を取り入れることで、開発環境におる課題（リソース確保にかかる工数、期間やコスト、環境保存、性能やセキュリティー等）を解決し移行を実現した株式会社 JSOL（以降、JSOL）の取組みである。JSOL は、顧客の課題解決や価値創出に向けて最先端のソリューションを提案している ICT サービスコーディネーターである。JSOL は、ホストコンピューターが主流の頃からインフラサービスも積極的に展開してきたが、時代の流れとともにオープン系システム、サーバー仮想化基盤、プライベートクラウドなどへとサービスの幅を広げていき、現在ではパブリッククラウドを高度に組み合わせたハイブリッドクラウド・ソリューション『J-Hive（ジェイハイブ）』に力をいれている。

本稿では、社内開発環境として導入されていたサーバー仮想化基盤がハードウェア保守サポート終了に伴う更改において、J-Hive を取り入れたハイブリッドクラウド構成にすることで、社内開発環境における課題を解決し、また移行をスムーズに実現した事例を紹介する。

2. 取り組みの目的

2.1. 解決すべき課題

JSOL の社内開発環境は、プロジェクト毎に立ち上げられ一昔前では、ビル内のサーバー室が物理サーバーで埋め尽くされるという状況であったが、2009 年以降は自社データセンターの仮想化基盤に集約され効率化が図られた。

また、開発部門においては物理サーバーを調達する必要がなくなったことで開発環境導入の障壁が低くなり、社内開発環境上の仮想サーバー台数も急速に増加していた。

この急速な増加とともに、社内開発環境は増強されたが長年の利用により様々な課題に直面していた。

課題１：最新の OS に対応した開発環境をタイムリーに提供する事が困難

Windows OS などの進化に合わせ、社内開発環境の仮想化ソフトウェア（VMware vSphere Server）は最適なパフォーマンスを発揮できるようにバージョンアップや機

[1] 事例提供：株式会社 JSOL　山本 勝 氏

能アップを続けており、常に最新の OS をサポートできる環境を構成するには、仮想化ソフトウェアのバージョンアップが必要であった。

しかしながら、仮想化ソフトウェアのバージョンアップに合わせて、関連する管理・運用ツールも全てバージョンアップを実施する必要があり、大掛かりな作業が必要となる。

課題２：リソースが上限に達した場合、タイムリーな環境提供が困難

社内開発環境がハードウェアリソースの上限に達すると、ハードウェアの増強が必要であるが、ハードウェア増強には承認プロセスや調達期間など数週間のリードタイムが必要となることもしばしば発生し、またハードウェア増強作業のためのコスト増加にもつながった。

課題３：最新のパブリッククラウドの機能を活用した開発ができない

ビジネスの変化や効率的な開発を実現するための新しい基盤として、パブリッククラウドが活用されている。

パブリッククラウドでは、ビジネスニーズに対応するためのサービスや機能が次々とリリースされているためだ。しかしながら、オンプレミスの開発環境だけでは、パブリッククラウドの最新機能を活用した開発が実施できない。

2.2. 目標

上記の課題を解決するために、新社内開発環境はオンプレミスとパブリッククラウドを組み合わせたハイブリッドクラウド環境とすることで、それぞれのメリットを取り入れることとした。

ただし、社内開発環境の抱えている課題（最新の OS の活用・タイムリーなリソースの利用・最新のパブリッククラウド機能での開発）を解決するために多大なコストをかけてしまっては、開発コスト自体の増加となることから、必要以上の投資とならないようにすることが必須とされていた。

このため、比較対象として最新のオンプレミス環境のみでの更改を実施した場合のコストを試算し、これに対してハイブリッドクラウド環境がコスト高とならないことを目標とした。

3. 取り組みの対象

3.1. 移行範囲

更改対象の社内開発環境は 2009 年から利用されていることもあり、対象は数百台に上ったが、長く利用されていないサーバーや、開発するために必要なサービスを提供するサーバーなどはこのタイミングで整理（削除や別環境への移行）を実施することとした。

3.2. 新社内開発環境の技術

（1）仮想化ソフトウェア

仮想化ソフトウェアは、従来から利用していた VMware vSphere Server を選択した。運用担当への運用オペレーションの変更による負荷がかからないことと、保守担当の技

術ノウハウが引き続き活用できると考えたからだ。

(2) ストレージ

ストレージは、以下の理由から、NetApp 社の FAS ストレージを選択した。

- 更改前の社内開発環境で利用しているストレージも NetApp 社の FAS ストレージであり、移行については、SnapMirror という機能でディスクのボリュームを遠隔地に非同期でレプリケーションできる。また、レプリケーション帯域も制御でき、業務に影響を与えない範囲でデータ移行が可能。
- ディスク容量を削減するための重複排除機能が備わっており、仮想化基盤ではその効率が比較的良い（仮想サーバーの OS 部分の重複排除効率が良いため）
- 仮想化ソフトウェアと連動して、ストレージのバックアップを取得できるバックアップソフトウェア（Snap Manager for VMware Infrastructure）の運用オペレーションが容易であり、運用担当が利用するための教育が不要であった。
- NetApp 社の FAS ストレージの OnCommand Shift 機能により、VMDK 形式（VMware vSphere Server での仮想サーバーのイメージファイル形式）を VHD 形式（マイクロソフト社の Hyper-V や Azure での仮想サーバーのイメージファイル形式）に高速に変換することが可能。
- NetApp 社の FAS ストレージをパブリッククラウドサービスから利用するソリューションとして、NetApp Private Storage for Cloud が提供されており、パブリッククラウドからのストレージ利用をサポート。

(3) パブリッククラウド

社内開発環境の移行先として、マイクロソフト社の Azure を選択した。開発環境としては、アマゾンウェブサービス社の AWS も利用しているが、既存環境の移行に関しては、NetApp 社の FAS ストレージの変換機能を用いることで、移行を高速かつ自動化できるためだ。

(4) パブリッククラウドへの接続

パブリッククラウドへの接続は、エクイニクス社が提供する Equinix Cloud Exchange サービスを利用した。これにより、Azure にも AWS にも 1 本の物理線で接続が可能となるためだ。

4. 取り組みの実施

4.1. 導入までのプロセス概要

本書では、導入までのプロセスとして、企画フェーズから移行フェーズまでを対象とし、各フェーズにおいてのポイントを紹介する。

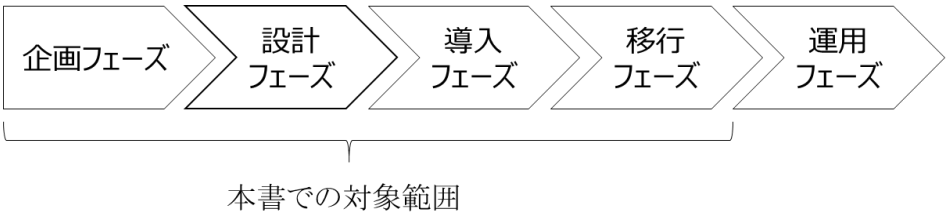


図 67-1

4.2. 企画フェーズ

そもそも、新社内開発環境は費用対効果があるかどうかが大前提であるため、既存環境全てをオンプレミス環境に移行した場合と、ハイブリッドクラウド環境へ移行した場合の試算を実施した。

試算段階では、台帳と仮想化基盤から収集したリソースデータを元にコストを算出した。

手順としては、パブリッククラウドへの移行可能な OS の仮想サーバーをピックアップし、仮想化基盤上での CPU やメモリ、ディスクの割当リソースから、Azure 上で必要なリソースを算出。これに、利用時間を当てはめることでパブリッククラウドのコストの算出が可能となる。

それに、パブリッククラウドへ移行できない仮想サーバーが収まるサイジングをしたオンプレミスの仮想化基盤のコストをプラスすることで、ハイブリッドクラウド環境でのコストを算出した。

パブリッククラウドのコストインパクトが一番高いのは、CPU とメモリの利用時間であるが、ほとんどの開発環境は、既に保守フェーズに入っており、保守対応における一時的な稼働であることが想定できるため、年間ベースで 2 週間ほどの稼働で算出した。

また、開発フェーズ中の開発環境はオンプレミス環境に余裕を持たし、そこに吸収し、保守フェーズに入った後で、パブリッククラウドへ移行する方針とした。

この結果、ハイブリッドクラウド環境でのコストはオンプレミス環境を下回り、費用対効果を出せると判断しプロジェクトをスタートした。

4.3. 計画フェーズ

試算段階では、移行対象の絞込みは実施していないが、計画フェーズでは長く利用されていないサーバーや、開発するために必要なサービスを提供するサーバーを洗い出し、削除や別環境への移行することで、全体コストを下げることから着手した。

（1）現状ヒアリング 1 回目

開発部門へ開発サーバーのクラウドへの移行可否についてアンケートを実施したところ、ほとんどが不可という回答となった。これは、パブリッククラウド利用への不安感や移行時の開発部門の負荷などが原因であった。

これに対しては、別途説明会を実施することで開発部門を安心させるとともに、新開発環境では、技術的に移行可能な開発環境は移行必須であるという移行方針を伝えることとした。

また、ここでは削除対象となる開発環境を絞ることに成功したため、移行台数は約

200 台であることがわかり、移行方針からオンプレミスへの移行台数は約 50 台、パブリッククラウドへの移行台数は約 150 台と算出した。

（2）現状ヒアリング２回目

当初、設計や運用など全てが決まってから説明会を実施する予定であったが、1 回目のアンケートが不完全で終わったことから、ある程度設計できた段階で新開発環境の説明会を開催し、移行方針や新開発環境の概要を知ってもらった上で、2 度目のアンケートを実施した。

これにより、パブリッククラウドで稼働させると逆にコストが高くなる開発環境を抽出し適切な移行先を決定した結果、オンプレミスへの移行台数は約 100 台、パブリッククラウドへの移行台数は約 100 台となった。

4.4. 設計フェーズ

設計フェーズよりプロジェクト体制は、オンプレミスチーム、Azure チーム、移行チーム、運用チームと４つに別れた。チーム毎に概要設計から詳細設計まで実施するが、概要設計のネットワーク部分に関しては、ハイブリッドクラウド構成で最もチーム連携が必要な部分であるため、各チーム内での概要設計とせず、全チーム横断での概要設計を実施した。

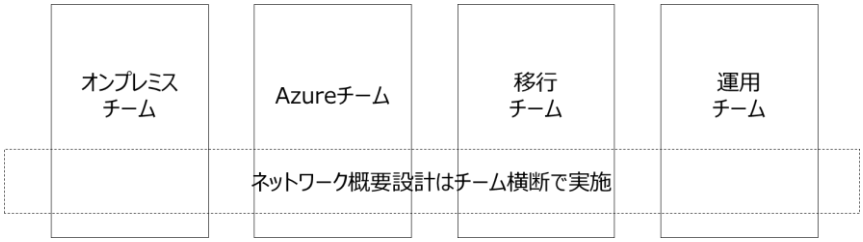


図 67-2

4.4.1. 全体構成

新社内開発環境は、JSOL のハイブリッドクラウド・ソリューション（J-Hive）を活用し、パブリッククラウドとオンプレミスの仮想化基盤を１つの開発環境として構成した。

移行に関しては、今回の移行に使う技術的な仕様から移行先は、Azure とし、AWS は新規に開発環境が必要な場合に選択できるようにした。

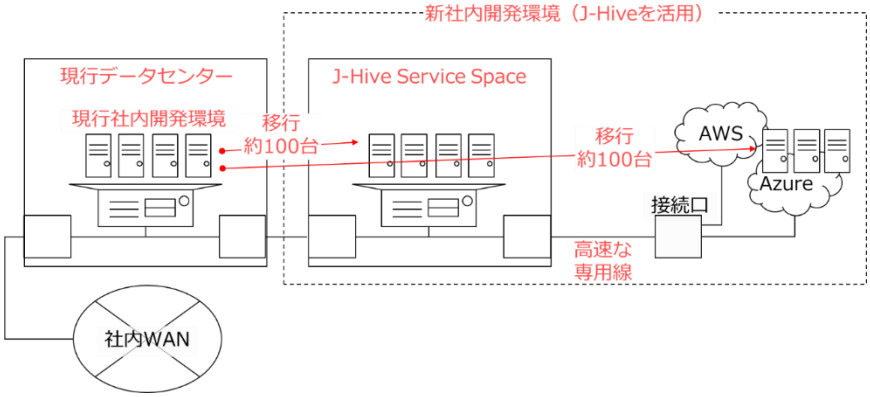


図 67-3

4.4.2. 仮想化基盤（オンプレミス）設計のポイント

（1）NetApp PrivateStorage for Cloud

仮想化基盤の設計については、従来の仮想化基盤の設計と物理面では特異な箇所はないが、パブリッククラウドへの柔軟な対応と移行方式から NetApp の FAS8020 ストレージ中心の構成をした。

従来のオンプレミスでのデータの保全や活用に関する手法は、ストレージ技術と密接に関わっており、今回の新社内開発環境においても、その保全や活用方法が継続的に活用できることを確認する必要があった。

例えば、データのバックアップは、ストレージのスナップショットにより瞬時に取得する事ができ、また瞬時に戻す（リストア）ことも可能である。開発環境においては、本番データをテストとして何度も利用することがあるが、このデータを利用する際に、ストレージの機能を活用することで、瞬時にデータの複製でき、利用後は一瞬で削除することや、利用前の状態に戻し、再度テストすることも可能である。

パブリッククラウドへ社内開発環境の比重を移す方針ではあるが、このデータ保全や活用の機能を引き続き利用するため、NetApp 社の FAS ストレージで提供されている NetApp PrivateStorage for Cloud ソリューション（以下、NPS）を採用した。柔軟に CPU やメモリを拡張できるパブリッククラウドからオンプレミスのストレージを直接接続することをサポートするからである。

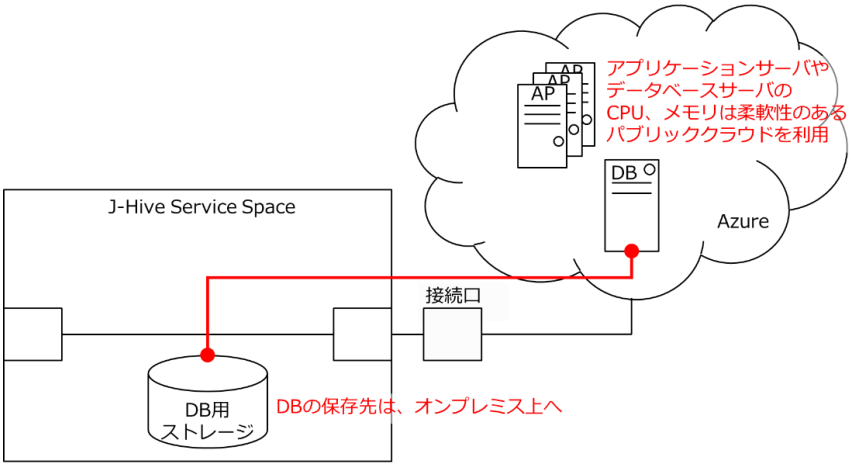


図 67-4

この構成については、事前に ERP パッケージでの業務利用が可能かどうかを検証し、オンプレミスで構成した場合と遜色なく稼働することが確認できていた。

表 67-1

#	環境	ERP 処理値	オンプレ環境を100とした場合の性能結果			
			CPU	メモリ	ディスク	業務処理
1	オンプレ環境	3,000	100	100	100	100
2	Azure+NPS環境	3,000	70	110	122	87

※ ERP 処理値 3000 での業務処理を実行、オンプレミス環境を 100 とした場合に、Azure ＋NPS 環境での能力を測定した。

（2）ネットワーク分離

新しく追加した構成はネットワークの分離だ。パブリッククラウドとオンプレミス上の仮想サーバーを通信させるには、社内のセキュリティー規定から、通常の社内 OA 環境で利用する社内ネットワークとは別のネットワークで構成することが必須条件であったためである。（旧社内開発環境は社内ネットワーク内に設置されている）

社内ネットワークと開発ネットワークの分離は、仮想化基盤内にファイヤーウォールを設置することで実現し、開発上ファイルを社内ネットワークから開発ネットワークに移動するための手段として、中間セグメントにファイル共有システムを準備した。

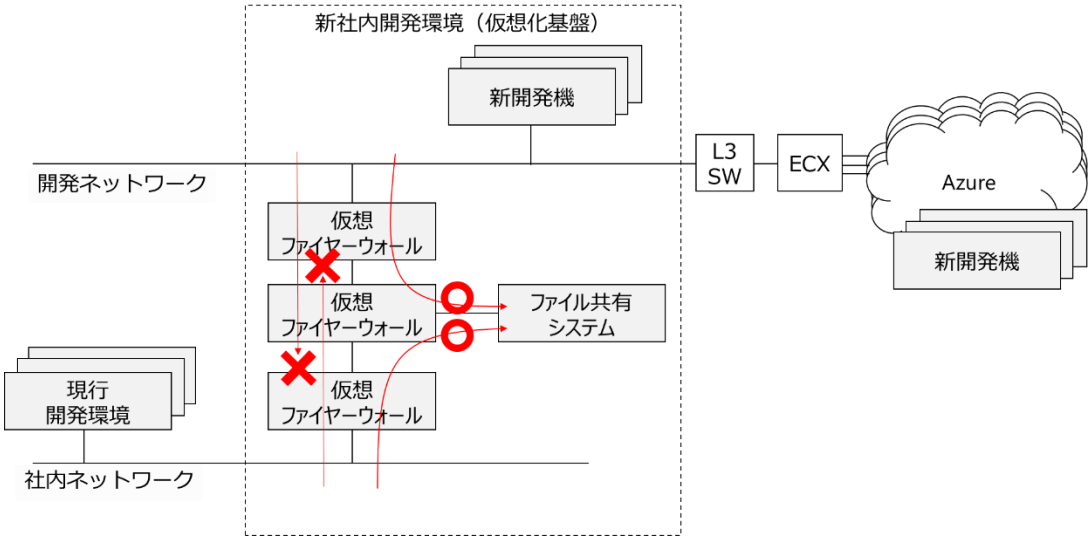


図 67-5

4.4.3. Azure 設計のポイント

（1）Equinix Cloud Exchange 設計

パブリッククラウドへの接続は、オンプレミスサーバーとの連携に遅延が発生しないように Equinix 社の Equinix Cloud Exchange（以下 ECX）による専用線接続を採用した。これによりセキュアな閉域環境での接続と安定した回線品質を確保できる。

JSOL で用意した L3SW（レイヤ 3 ネットワークスイッチ）から ECX まで 10Gbps で

物理的に接続し、パブリッククラウドへは、この物理接続内に複数の論理接続を構成し接続先のパブリッククラウドを柔軟に増減できる。

この機能を利用することで、激しく増減する開発環境のネットワークを適切なタイミングで作成・削除が可能となった。また、L3SW には、仮想の BGP ルータを複数作成する事が可能な Cisco 社の Nexus を利用することで L3SW 側も作成・削除が適切なタイミングで実施できる。

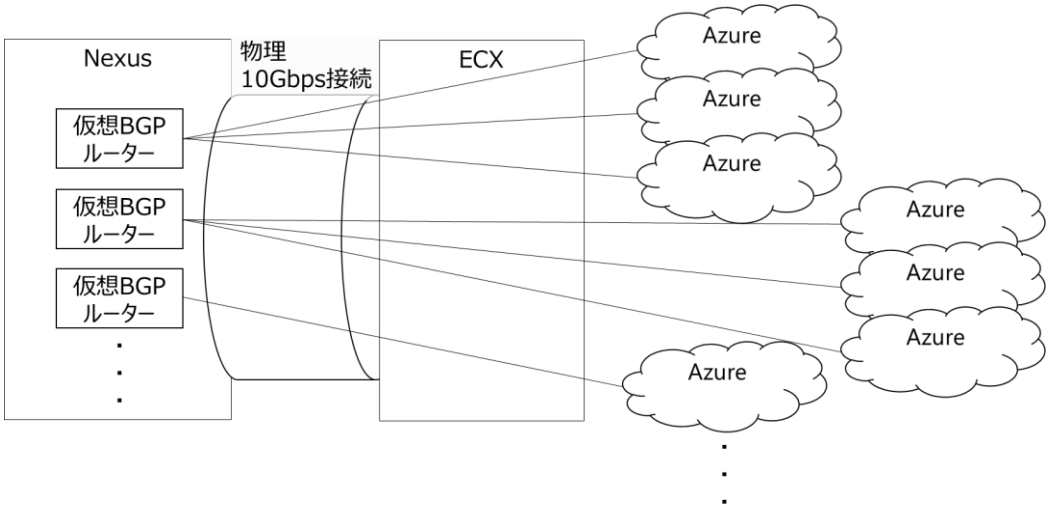


図 67-6

（2）課金設計

開発環境は、各業務・プロジェクトへ課金が必要となるため、Azure の利用料金が各業務・プロジェクト毎に把握できる仕組みが必要であった。

旧社内開発環境はオンプレミスのみの環境であるため、CPU・メモリ・ディスクの単位を決め、サーバー１台あたりの課金額を算出していたが、Azure の利用明細はサーバー１台あたりでの課金算出が困難な形となっており、新たな課金の仕組みとして、サブスクリプションと呼ばれる単位で課金算出することとした。

１サブスクリプションには、複数のネットワーク、複数の開発環境を作成する事ができるが、運用負荷を軽減させるために、「１サブスクリプション：１ネットワーク：１開発環境」という基本パターンを決定した。

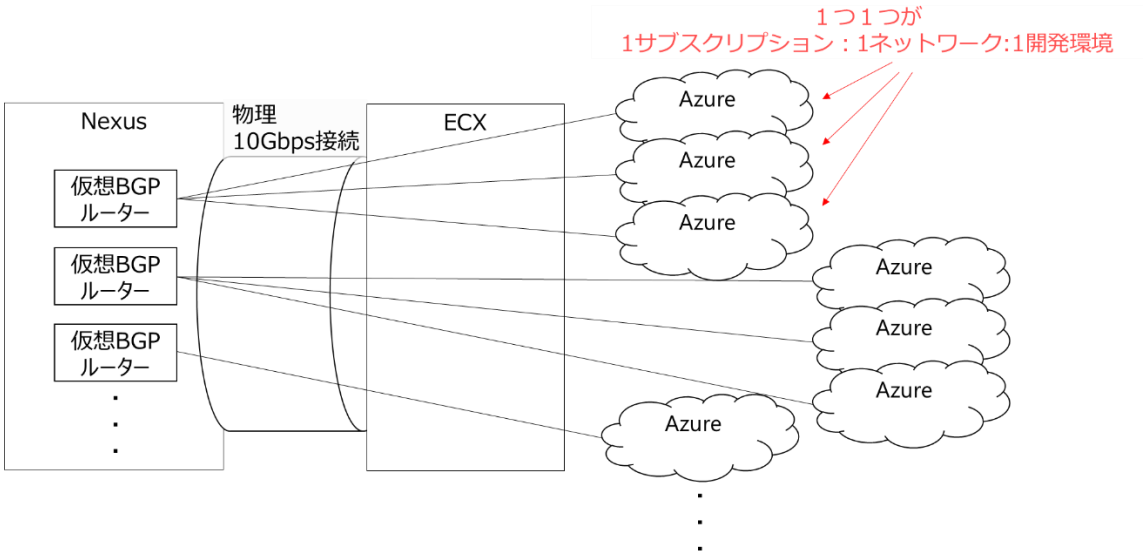


図 67-7

（3）インターネット接続設計

Azure 上の仮想サーバーはインターネットに自由に接続できる環境が提供される。しかしながら、JSOL の規定ではインターネットへの接続については、十分なセキュリティー対策を取った上での接続が必須条件であり、これは社内開発環境の仮想サーバーでも同様である。

また、仮想化基盤（オンプレミス）設計で述べたように、社内ネットワークと開発ネットワークを分離することから、今まで社内ネットワークで提供されてきたセキュアなインターネット接続環境は利用できない。

このため、Azure 上の仮想サーバーがインターネットに自由に接続できることを制限し、セキュアなインターネット接続環境を社内開発環境用に提供するという 2 つについて、設計する必要があった。

まずは、自由なインターネット接続の制限だが、こちらについては、Azure ネットワークセキュリティー機能として、ネットワークセキュリティーグループ（NSG）というステートフル動作のレイヤー 4 レベルパケットフィルタリング機能を適用した。

Azure ネットワークセキュリティー構成方針を以下のとおり。

- Azure 環境から直接インターネットへの通信を遮断する。
  （※ただし Azure サービスとして必要な通信がある場合は IP アドレスおよびポートを絞って許可とする）
- サブネット単位に NSG を構成および適用する。

基本的に Outbound ルールで制限を行い、Inbound ルールでの制限は行わない。

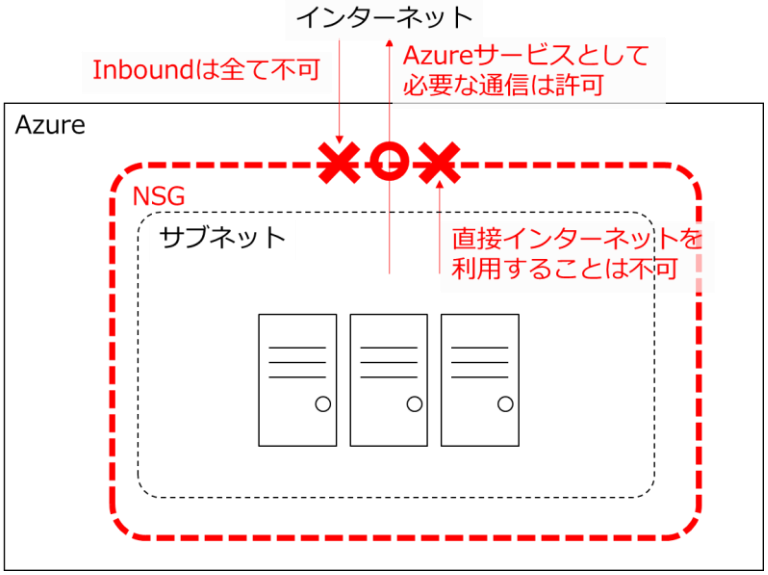


図 67-8

続いて、インターネットへの接続の提供だが、開発環境用に新たなインターネット接続回線は設けずに、Azure からのインターネット利用を基本に構成した。ただし、前述のとおり Azure からのインターネット利用は、セキュリティー対策が従来の社内ネットワークのセキュリティー対策と同様の対策が取られていないため、仮想ファイヤーウォールが備えているセキュリティー対策を経由させる方式とした。

仮想ファイヤーウォールのインターフェイスアドレスをプロキシサーバーのアドレスとして仮想サーバーに設定することで、仮想サーバーは仮想ファイヤーウォールのインターフェイスを経由する。この時に仮想ファイヤーウォールが備えている、ウィルス対策機能、WEB フィルタリング機能が適用され、インターネットへのセキュアな接続が実現する。

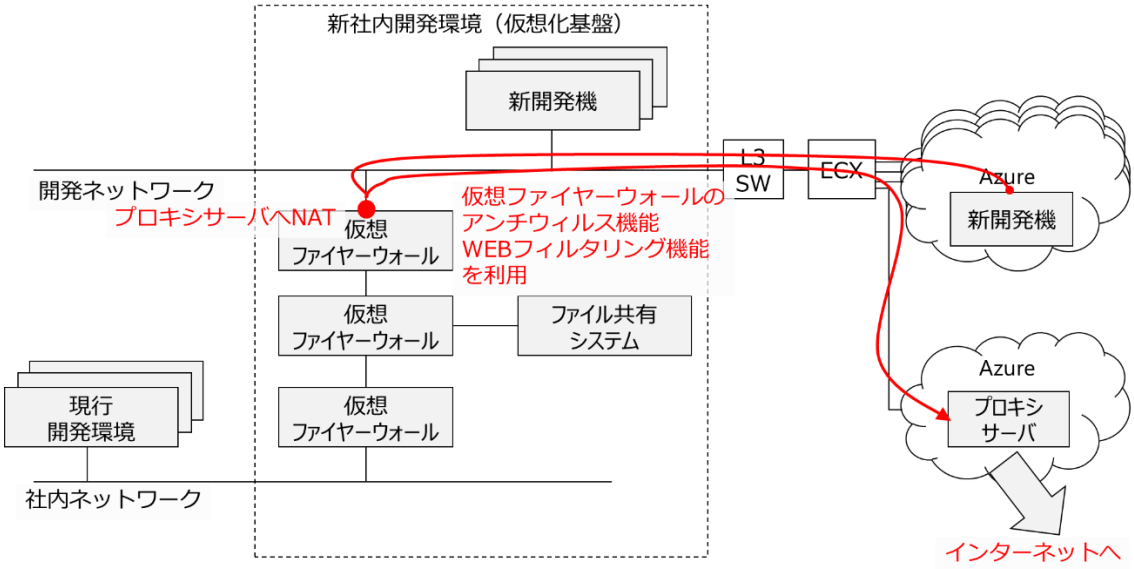


図 67-9

（4）Azure Automation 設計

Azure Automation とは、Azure 上での様々な作業を自動化させるために利用できる。この機能で以下を実現した。

- 仮想サーバー全停止
  - ✓ 各サブスクリプション内の仮想サーバーは、課金量を抑制するために毎日指定時間にシャットダウンを行う。
  - ✓ サブスクリプション内の仮想サーバーのシャットダウンは基本的には全て、Azure Automation 機能にてスケジュール実行を行う。
  - ✓ 実行間隔は毎日 20:00 とする。
  - ✓ 除外したい対象仮想マシンを指定したい場合は個別設定できる方式とする。
- 仮想サーバー全開始
  - ✓ 全停止した仮想サーバーの自動開始をサポートする。
  - ✓ サブスクリプション内の全仮想サーバーの開始は Azure Automation 機能にてスケジュール実行を行う。
  - ✓ 本機能は、利用者が必要に応じて設定するものとし、管理者の初期設定作業には含めない。
  - ✓ 除外したい対象仮想サーバーを指定したい場合は個別に設定できる方式とする。
- バックアップ取得
  - ✓ サブスクリプション内の全仮想サーバーのバックアップ取得は、Azure Automation 機能によるスケジュール実行を行う。
  - ✓ 取得間隔は毎日 4:00 とする。
  - ✓ バックアップ取得は仮想サーバー停止されている状態で取得するようにスケジュール設定する。
  - ✓ 除外したい対象仮想サーバー指定したい場合は個別に設定できる方式とする。
  - ✓ Azure Automation で実行するプログラム内で世代管理を行い、過去最大 3 世代分のバックアップが存在する形で古いイメージを自動的に削除する。

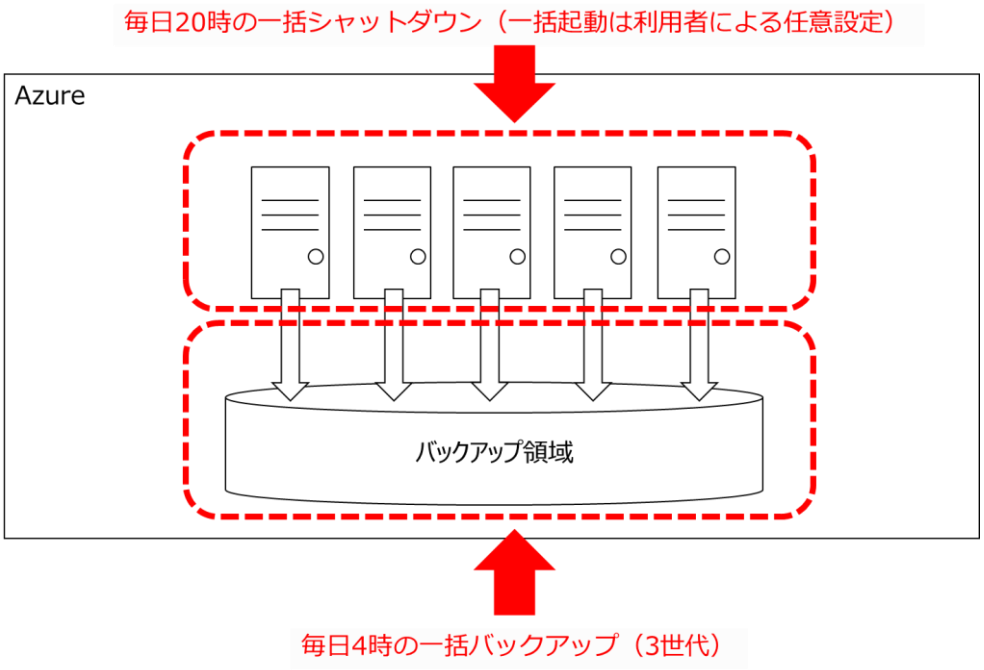


図 67-10

仮想サーバー全停止、仮想サーバー全バックアップは、管理者にてスケジュールも含めた初期設定を行うが、細かいスケジュール調整は、利用者が柔軟に変更できる構成とした。

4.4.4. 移行設計のポイント

今回採用している NetApp 社の FAS ストレージには、OnCommand Shift という機能により仮想サーバーのイメージを VMDK 形式、VHD 形式、通常の Volume 形式変更することができる。

イメージの変換は数十秒という高速で変換される。従来のソフトウェアによる変換はイメージの容量が大きければ数時間かかる場合もあったが、この機能は容量に関係なく高速である。

また、Azure および NetApp 社の FAS ストレージはパワーシェルでの実行が可能なため、イメージ変換、Azure へのアップロード、Azure 上での諸設定を自動化する移行方式とした。

以下は、本プロジェクト前に移行方式について検討した結果であるが、30GB の仮想サーバーイメージを NetApp 社の FAS ストレージ内で、イメージ変換し Azure へアップロードするまでを自動化している。アップロード自体は通常のコピーであるため、時間としては一番かかるところではあるが高速な専用線を用いた J-Hive でのハイブリッドクラウド構成により安定した速度でのアップロードが可能であった。

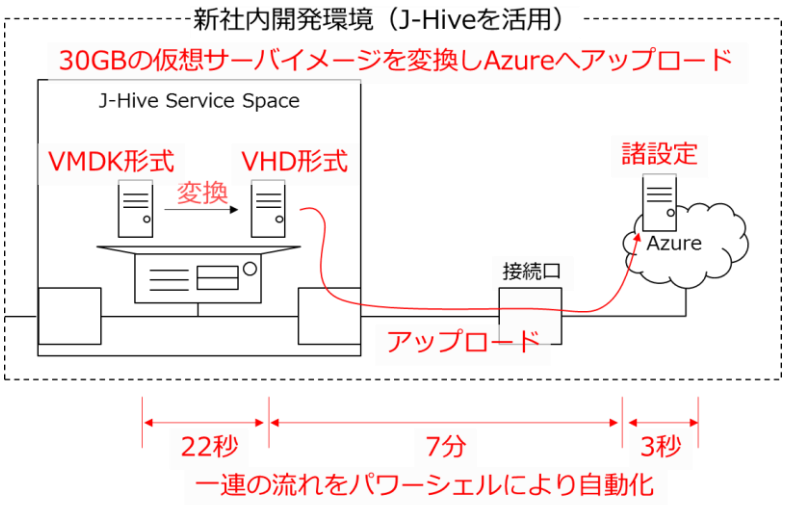


図 67-11

実際の移行パターンとしては、オンプレミスからオンプレミスへの移行とオンプレミスから Azure への移行と２パターンであるが、２つの移行のどちらとも、一括移行にしてしまうとストレージボリューム単位の移行となってしまう。

一括移行の場合は、移行が完了するまでそのストレージボリュームに配置されているシステムは起動することができない。１つのストレージボリュームには、多くの仮想サーバーが集まっており、停止時間が長くなってしまう。

このため、仮想サーバー１台単位で移行できるような設計を実施する必要があった。

（1）オンプレミスからオンプレミスへの移行

旧社内開発環境のストレージは NetApp 社の FAS ストレージであり、NetApp 社の FAS ストレージ同士であれば SnapMirror 機能によりストレージボリューム単位でレプリケーションが容易に実行できる。

ただし、オンプレミスへ移行する仮想サーバーと Azure へ移行する仮想サーバーが同じストレージボリューム内に混在することと、移行先の新社内開発環境のストレージボリューム構成が旧社内開発環境と違うことから、単純に SnapMirror でレプリケーションをすることはできでない。

このため、いったん SnapMirror で旧社内開発環境から新社内開発環境にレプリケーションしてから、必要な仮想サーバーのみを最適なストレージボリュームに新社内開発環境内で移動（コピー）させる手段をとった。

（2）オンプレミスから Azure への移行

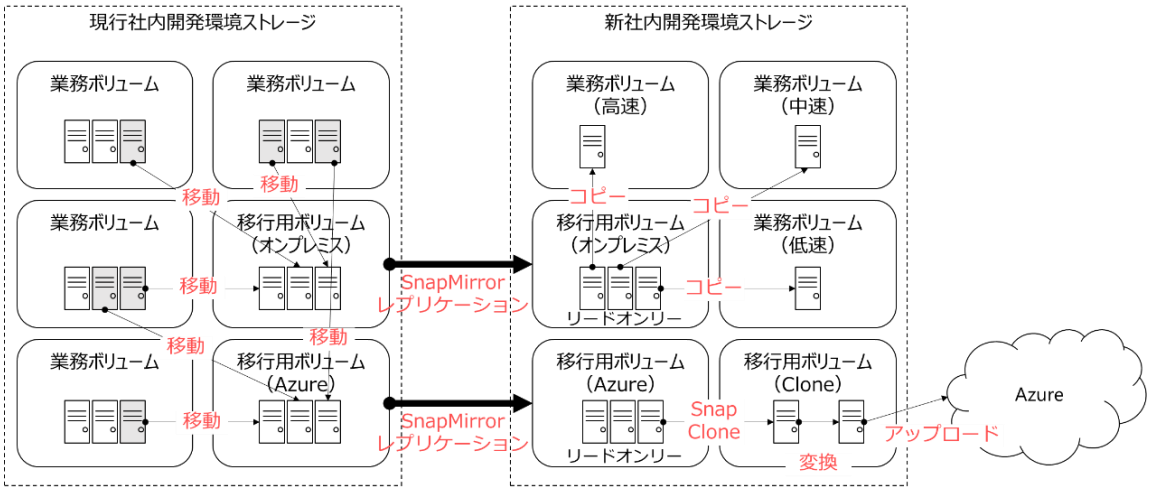
Azure への移行に関しては、旧社内開発環境から新社内開発環境のストレージまで SnapMirror を利用して移行し、そこから別方式により Azure へ移行する。

SnapMirror は、レプリケーション先（今回は、新社内開発環境のストレージとなる）のストレージボリュームはリードオンリー（読み取り専用）となるため、仮想サーバー

のイメージを変換する機能を適用することができない。

そのため、SnapClone という複製技術を利用し、読み書き可能なストレージボリュームにコピーする。実際は元のストレージボリュームの変更点のみ書き込まれるという方式のため、高速でディスク容量も削減できる。

その後、前述で述べた OnCommand Shift 機能でイメージ変換、自動アップロード・自動設定で移行を実行する。



※同じストレージ筐体内での移動、コピーは比較的高速に実施可能。

※SnapClone に関しては数秒という超高速で実施可能。

図 67-12

4.5. 導入フェーズ

導入においては、仕様どおりの構築、テストでの確認を基本とするが、パブリッククラウドにおいては、設計時と導入時ではパブリッククラウド自体のアップデートにより仕様変更がされてしまう場合がある。

このアップデートは、不具合が改善されることがあるが、制限事項として考慮し設計していたものが制限事項でなくなるといった場合もあり、それを取り入れるかどうかの検討と、取り入れる場合は設計の改訂が必要となってくる。

本プロジェクト中においても、アップデートによる以下の仕様変更が発生し、都度設計について再検討や見直しを実施した。

（1）Azure のリリース

旧社内開発環境が関西圏に設置されているため、関西圏の Equinix から Azure への接続を検討していたが、関西圏の Equinix から Azure への専用線接続のリリースが未定のため、関東圏の Equinix からの接続でプロジェクトを進めていた。

しかし、構築中に関西圏からの専用線接続が急遽リリースされたため、移行時間の短縮などを考え、導入中の設備を関東圏から関西圏に移設した。

(2) デプロイモデルの変更

設計時点で、すでにデプロイモデルがクラシックデプロイモデルとリソースマネージャデプロイモデルの２パターンあり、将来的にはリソースマネージャデプロイモデルに統合されていくことが予想された。

しかし、リソースマネージャデプロイモデルでの専用線接続の実装事例もないため、新社内開発環境ではクラシックデプロイモデルでの実装とし、運用中にリソースマネージャデプロイモデルへ切り替える方式とした。

(3) Azure の仮想サーバーのネットワークカード（以下 NIC）の制限

本プロジェクト前の検証段階においては Azure では NIC カードが 1 枚のみのサポートであったが、本プロジェクトが開始前のアップデートで複数 NIC がサポートされていた。

導入段階で複数 NIC の自動設定について再度検討したが、企画フェーズでの調査で複数 NIC の仮想サーバーが少なかったため、手動での個別対応とした。

4.6. 移行フェーズ

移行フェーズでは、当初の移行アンケート結果を元に再度ヒアリングし、移行先の最終決定、移行スケジュールを決定した。

移行先の最終決定とは、オンプレミスへの移行か Azure への移行かを、当初アンケートをとった際には確認できなかった特殊な利用方法や、開発環境で利用しているアプリケーションのライセンスがクラウドに移行できないものなど、移行要件を確認し最終決定した。

移行スケジュールについては、移行単位をストレージボリューム単位ではなく、仮想サーバー単位で実施する方式としているため、長い時間のシステム停止は発生しないものの、移行後の正常性確認まで実施すると営業日内でシステムを停止する必要があった。

このため、ヒアリングの際に停止可能な期間を確認していくことになったが、開発ピークであるプロジェクトのシステムについては、移行を 1 ヶ月以上後ろに倒す必要があるものもあった。

当初、移行スケジュールは 6 ヶ月の期間内で完了を予定していたが、上記のような調整も複数発生し、移行完了までに 8 ヶ月の期間がかかった。

また、この移行フェーズにおいて、最も苦労した点は、移行後の Azure 環境利用における問合せ対応であった。

旧開発環境の管理コンソールは利用者に公開していなかった。これは、管理コンソールでの作業が管理者により実施していた仮想サーバーの作成が主たる操作のためである。

しかし、Azure においては仮想サーバーを作成するだけではなく、PaaS と呼ばれるアプリケーションを構成するパーツを利用するなど、いくつものサービスを今後利用していく必要があった。

このため、従来どおり管理者のみがコンソールを利用する方式では、パブリッククラウド上の膨大な PaaS を開発用として作成するノウハウを維持することは困難であり、運用負荷も懸念の一つとなっていた。

逆に、利用者（開発をする SE）は、パブリッククラウドの活用ノウハウを習得しないと、パブリッククラウドでの開発力を強化できない可能性があるため、結果 Azure の管理コンソールは利用者に公開することになった。

以上より、利用者は仮想サーバーの新規作成やその際に発生するネットワーク等の諸設定を実施する必要があり、移行後に利用を開始したなれない利用者からの問合せが多く発生し、管理者は対応に追われた。

5. 取り組みの結果

移行期間はスケジュール調整の結果、当初の予定内では完了しなかった。しかし、今回 NetApp 社の FAS ストレージを採用したことで NetApp 社の FAS ストレージを採用しなかった場合と比較すると、２分の１の作業時間、３分の１の作業者の工数での移行作業で実施ができた。以下がその主な理由である。

- オンプレミスからオンプレミスへの SnapMirror による差分コピーの移行
- SnapClone によるイメージ変換の用の仮想サーバーの高速なコピー
- OnCommnad Shift による高速なイメージ変換、パワーシェルによる一連の作業の自動化による作業体制の簡素化（スクリプトの実行のみであるため、１名での作業が可能）

また、Azure へ移行した開発環境に関しては、旧社内開発環境に比べ高コストとなるものと低コストになるものに分かれた。これは、プロジェクトが完了し、安定した保守フェーズに入っている開発環境であれば、システムを停止する期間が長く低コストとなるが、保守フェーズでも改訂を繰り返しているシステムの開発環境であれば稼働時間が長くなり、20 時の一斉シャットダウンも適用していないためだ。

これを改善するために、オンプレミスと Azure 間で開発環境の特徴に合わせて移動できるような処置を取り、コスト最適化を実現した。

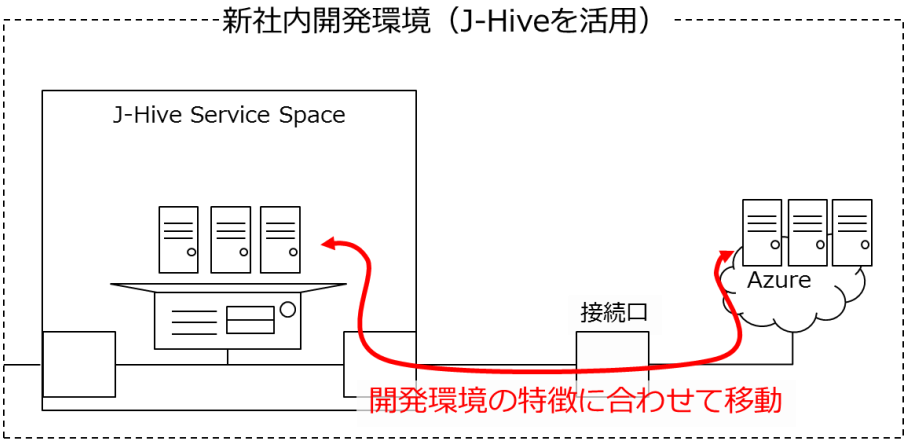


図 67-13

また、パブリッククラウドを開発環境に取り込んだことにより、先に課題として挙げた

・最新の OS に対応した開発環境をタイムリーに提供する

・リソースが上限に達した場合などでのタイムリーな環境提供

・最新のパブリッククラウドの機能を活用した開発

が可能となった。

6. 今後の取り組みと考察（今後の課題）

今後は、パブリッククラウドでのシステム構築は技術的な視点だけなくコスト最適化の視点での開発が重要であると考える。例えば、夜間のバッチ処理でシステムと連携するのであれば関連する全てのシステムを起動しておく必要があり、パブリッククラウドでは高コストになる。

低コストでの稼働を考えるのであれば、夜間バッチ処理ではなくシステムのサービス時間中にリアルタイムで連携し、サービス時間はシステムを停止するといった仕組みを導入する必要がある。

また、パブリッククラウドのシステム利用が将来進んでいくと、各パブリッククラウド間でデータの連携が複雑化し、利用しているパブリッククラウドを変更するには大量の変更作業の負荷が発生し、結果として連携の複雑さによりサービスにロックインされてしまうといったことが考えられる。

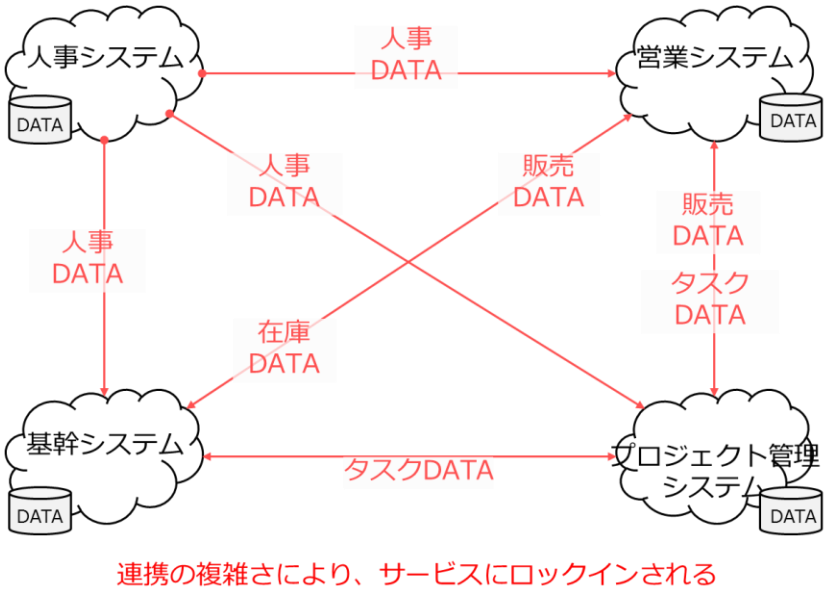


図 67-14

将来を見越してデータの連携を考えていくには、上記のようなロックインを避けるためにデータを中心部に置き、連携はデータ中心部とやり取りするといった構成を検討する必要がある。これにより連携が簡素化され、サービスにロックインされないだけでなく、テータ中心部だけ

が 24 時間 365 日稼働していれば、各システムは自身のサービス時間のみ稼働させておくことができる。

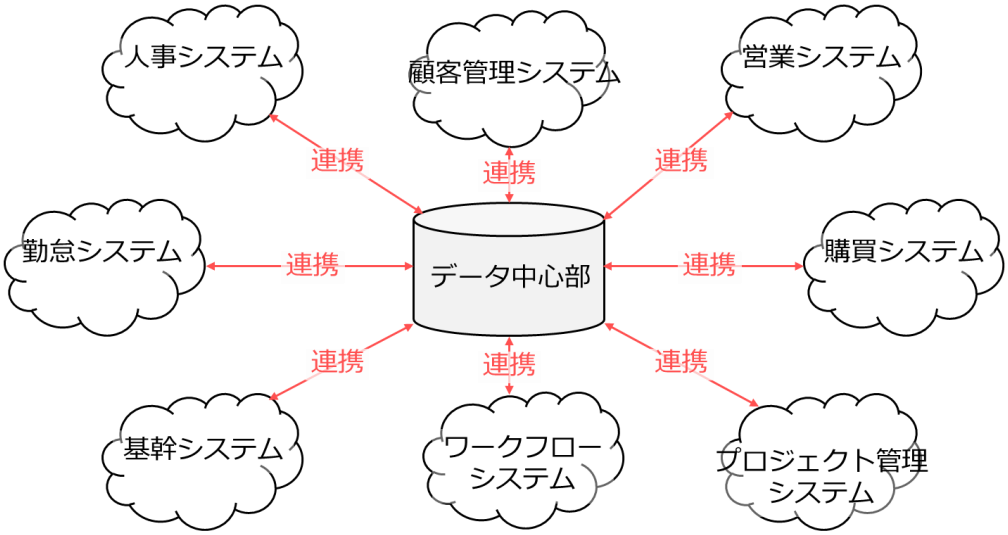


図 67-15

上記のような構成にするには、安全で高速にパブリッククラウドと連携することで安定した連携が実現できると考えており、今回の新社内開発環境で活用した J-Hive ソリューションのようなハイブリッドクラウド構成が重要なファクターであると考えている。

掲載されている会社名・製品名などは、各社の登録商標または商標です。

独立行政法人情報処理推進機構 技術本部 ソフトウェア高信頼化センター（IPA/SEC）