SQL Server テクニカル e-Book シリーズ

Microsoft

SQL Server による ビジネス 継続性の 促進

SQL Server による ビジネス 継続性の 促進

目次

本書について

本書では、データベース プラットフォームのコンテキストで一般的な業界固有の用語が頻繁に使用されています。取り上げる話題の目的上、以下の定義を使用します。

サーバーとは、オペレーティング システムを搭載し、SQL Server ソフトウェアをホストしている物理マシンまたは仮想マシンです。

クラスターとは、冗長性を提供するためにグループ化された一連のサーバーのことです。クラスター マネージャーがクラスター メンバー (ノード) の正常性と応答性を監視します。Windows Server では、マネージャーは WSFC (Windows Server フェールオーバー クラスター [マネージャー]) と呼ばれます。Linux ディストリビューションでは、Pacemaker をクラスター マネージャーとして使用します。クラスターは、オペレーティング システム レベルで管理されます。

ノードとは、フェールオーバー クラスターのメンバーです。クラスター内の各ノードが要求に応答する能力は同じで、クラスター内のすべてのノードが同じデータにアクセスできます。各ノードは、クラスター マネージャーが「正常」と見なすタイミングで要求に応答できる必要があります。

データベースとは、ディスク上の物理ファイルにアプリケーション データとメタデータを格納するデータ構造です。

インスタンスとは、特定時点で特定コンピューターにあるメモリで実行される単一の SQL Server サービスが実行する SQL Server データベース、ジョブなどのコレクションです。インスタンスは単一の IP アドレスを使用してアクセスを受け、すべての要求はその IP に送信されます。要求の送信元は、受信する応答の送信元となる物理的な場所は把握しません。

レプリカ (またはデータベース レプリカ) とは、1 つのフェールオーバー構成でグループ化された 2 つ以上の SQL Server インスタンスで構成されるセットのことです。つまり、レプリカはデータベース レベルで、クラスターはオペレーティング システム レベルとなります。

本書の対象読者

この e-Book は、ミッション クリティカルなアプリケーションの稼働時間を維持することに関心を抱くデータベース アーキテクト、管理者、開発者の皆様を対象としています。この e-Book を参考にすると、Linux ベースと Windows ベースの両方のデプロイメントの機能と統合されたミラーリングおよびクラスタリングのテクノロジを利用して、高可用性とディザスター リカバリーの目標を実現する点で SQL Server がどのように役立つかを理解できます。この e-Book では、Always On フェールオーバー クラスター インスタンス、Always On 可用性グループ、ログ配布などのツールや機能について取り上げ、こうした機能を実践で使用するための方法に関する技術的詳細について記します。

私たちは、企業やエンド ユーザーがさまざまな方法で情報を利用したり生み出したりするデータ主導の世界に住んでいます。

いつでもどこにいても、データに 24 時間迅速にアクセスできることが期待されています。データの需要が高まり、世界経済が拡大するにつれ、途切れることなく迅速にデータにアクセスできるようにすることは、組織にとって重要な関心事となっています。アクセスできないと、顧客へのサービスの提供や、日々のタスクの実行に大きな影響が出ます。

ミッション クリティカルなアプリケーションを常に稼働させておくことは、常にオンラインになっている今日の世界で不可欠です。これを実現するには、計画外のダウンタイムを最小限に抑える必要があります。ダウンタイムの原因となるものはいくつかあります。たとえば、ハードウェア / ソフトウェア保守のアップグレード、ネットワーク停止または停電、ハードウェア障害、セキュリティ違反、サイバー攻撃 (ハッキング、ウイルスなどによる) などです。ダウンタイムを最小限に抑え、できるだけビジネスに影響が出ないようにするには、途切れることがほとんど、あるいは全くなくビジネスを継続的に稼働させるためのバックアップ ソリューションとディザスター リカバリー ソリューションを施行する必要があります。

IT 業界にいる人たちは、高可用性 (HA) とディザスター リカバリー (DR) の概念に精通しています。簡単に言えば、HA とは致命的な障害が発生した場合でもビジネスを継続させるソリューションの実装のことです。これは、ハードウェアまたはソフトウェアの障害の影響を隠し、アプリケーションの可用性を維持することで、ユーザーに認識されるダウンタイムを最小限に抑えます。DR とは、障害発生後のデータ損失を解決できるようにするソリューションの実装のことです。高可用性システムでは、より確実にデータにアクセスすることができます。しかし、データの正確性を維持するために DR ソリューションも実装したいと思うかもしれません。

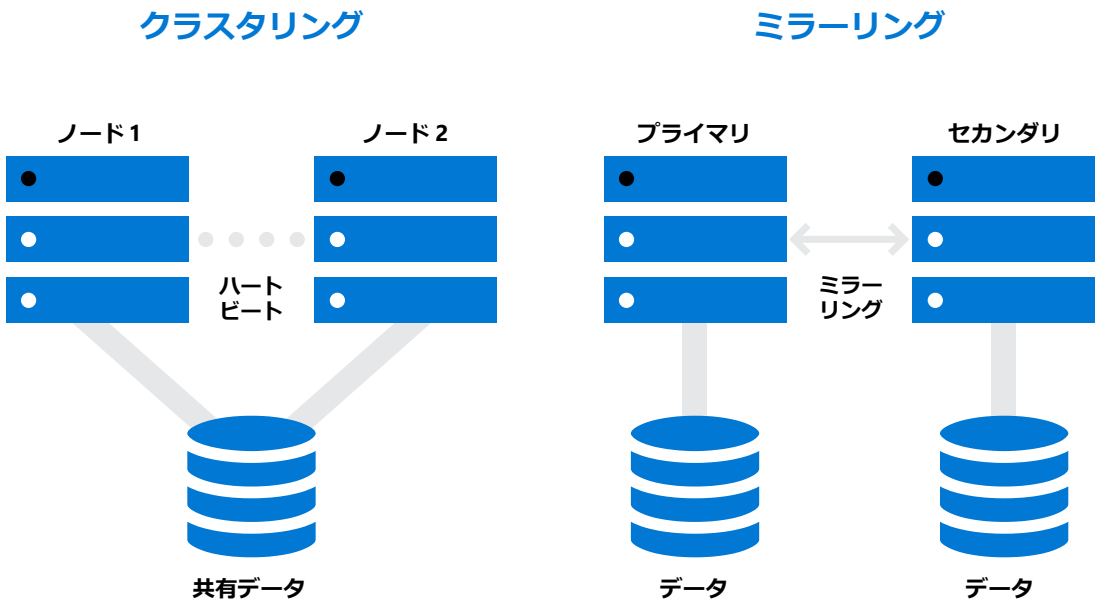
ミラーリングは廃止され、今後のバージョンの SQL Server では削除されます。Always On 可用性グループではこの機能が強化されているため、今後はぜひこちらを使用してください。

ビジネス継続性のための高可用性とディザスター リカバリー (HADR) ソリューション

高可用性とビジネス継続性を実現するための最も一般的な手法が 2 つあります。それは、ミラーリングとクラスタリングです。ミラーリングはデータベース レベルの継続性ソリューションです。クラスタリングはサーバー レベルのソリューションを提供します。

データベース ミラーリングは、別のハードウェア上にアクティブ データベースの完全なコピー (ミラー) であるスタンバイ データベースを維持することにより、ほぼ即時のフェールオーバーをサポートします。データベース ミラーリングは安全性の高い同期モードで運用できます。このモードの場合、受信トランザクションがすべてのサーバーに同時にコミットされます。またはパフォーマンスの高い非同期モードでも運用でき、その場合には受信トランザクションはアクティブ データベースにコミットされ、その後、(事前に選んだ特定時点で) ミラーにコピーされます。つまり、各ミラーで独自のデータソースを維持します。ミラーリングはデータベース レベルのソリューションで、完全復旧モデルを使用するデータベースでのみ動作します。

データベース クラスタリングは、(単一のデータ ストレージを共有する) 複数のサーバーを組み合わせてユーザーから単一のインスタンスに見えるようにするプロセスです。複数のユーザーがこのインスタンスに接続し、インスタンス内のどのサーバーがその時点でアクティブかを把握する必要はありません。1 台のサーバーで障害が発生したり、メンテナンスのためにオフラインにする必要が生じたりする場合にも、ユーザー エクスペリエンスには変化が生じません。クラスター内の各サーバーは、クラスター マネージャーがハートビートを使用して監視します。そのため、クラスター内のアクティブ サーバーがオフラインになったことを検出すると、クラスター内の次のサーバーにシームレスに切り替えようとします (切り替え時には、不定の遅延が生じます)。クラスター内のすべてのノードでデータが共有されるため、データの継続性が確保されます。 ■



Microsoft SQL Server は HADR をサポートするように設計され、重要なビジネス インフラ コンポーネントが常に稼動し、使用可能で、障害なく動作できることが保証されます。

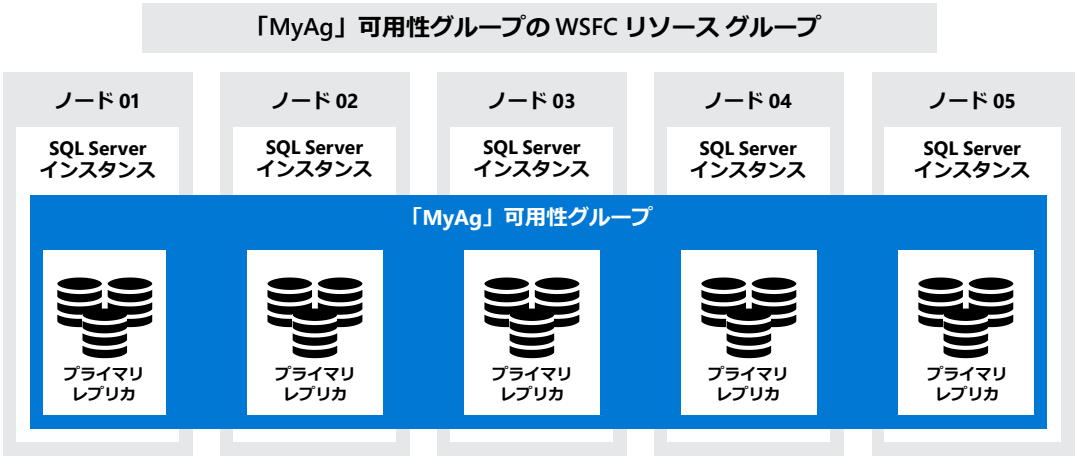
SQL Server にはさまざまな HADR シナリオ用のソリューションが備わっていて、組織が幅広い可用性に関する SLA を達成できるようにする一連の機能を提供します。SQL Server を使用すると、一層高いレベルのセキュリティ、信頼性、スケーラビリティを確保でき、クラスターやストレージ管理を向上させるさまざまなオプションが備わっています。Linux ベースと Windows ベースのどちらのデプロイメントでも利用できる Always On フェールオーバー クラスタリング インスタンス (FCI)、Always On 可用性グループ、ログ配布などの機能と、ミラーリングとクラスタリングのテクノロジを統合します。

インスタンス レベルの高可用性のための Always On フェールオーバー クラスター インスタンス

Always On FCI は、インスタンス レベルまたはサーバー レベルでビジネス継続性を提供する高可用性とディザスター リカバリーのツールです。FCI は、ネットワーク、ハードウェア、オペレーティング システム、またはソフトウェアの問題に起因するサーバー障害から保護します。FCI は単一のネットワーク アドレスを使用して要求を受信するため、要求を行うユーザーやアプリケーションは接続情報を変更する必要はありません。たとえば、FCI を使用するプライマリ サーバーが電力を失った場合、クラスター マネージャーは、クラスター内のセカンダリ サーバーの 1 つをプライマリ ポジションに単に「昇格」し、そちらに要求を再ルーティングします。応答しないクラスター メンバーは、オンラインに戻るまで要求を受け取りません。

データベース レベルの高可用性のための Always On 可用性グループ

Always On 可用性グループは、エンタープライズ レベルの高可用性とディザスター リカバリーのソリューションで、1 つ以上のユーザー データベースに対して最大の可用性を提供できます。可用性グループは、1 つのバックアップに共にフェールオーバーするユーザー データベースの 1 つ以上のグループです。



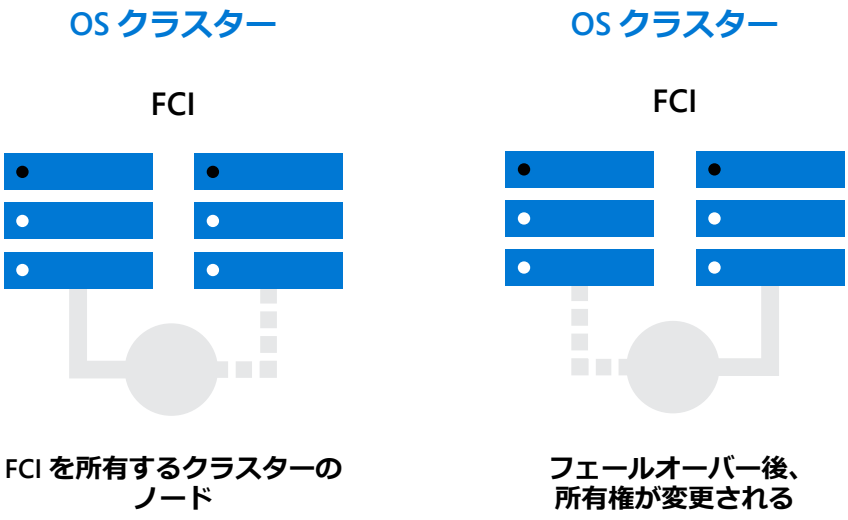
インスタンス レベルとデータベース レベルでのフェールオーバーの比較

Always On 可用性グループでは、SQL Server インスタンスがクラスター ノード上に存在する必要があります。SQL Server を使用するインスタンス レベルのフェールオーバーの場合、FCI は共有ストレージで依存関係を持ちます。対照的に、Always On 可用性グループを使用するデータベース レベルのフェールオーバーの場合、共有ストレージでは依存関係はありません。インスタンス レベルのフォールオーバーとデータベース レベルのフェールオーバーには、次の表に示されている概念的な違いがあります。

	インスタンス レベル	データベース レベル
クラスターの使用	**はい**	**はい**
保護レベル	**インスタンス**	**データベース**
ストレージの種類	**共有**	**非共有** 可用性グループ内のレプリカはストレージを共有しませんが、FCI がホストするレプリカは、その FCI が必要とする場合に共有ストレージ ソリューションを使用します。ストレージ ソリューションは、FCI 内のノードによってのみ共有され、AG レプリカ間では共有されません。
ストレージ ソリューション	**直接接続、SAN、マウント ポイント、CIFS**	**ノード タイプによって異なります**
フェールオーバー リソース	**サーバー、インスタンス、データベース**	**データベースのみ**

FCI は、インスタンスとして知られる SQL Server のインストール全体を使用可能にするための実証済みの方法です。

フェールオーバー クラスターは、インスタンスの 2 つ以上の同一のコピーで構成されます。つまり、データベース、SQL Server エージェント ジョブ、リンクされたサーバーなど、インスタンス内のすべての重複コピーが各ノードに存在します。一度にアクティブにできるノードは 1 つだけです。その他はパッシブになります。アクティブ サーバーで問題が発生した場合、インスタンスの所有権がパッシブ サーバーの 1 つに移動します。すべての FCI には、ネットワーク経由で提供される場合でも何らかの共有ストレージが必要です。

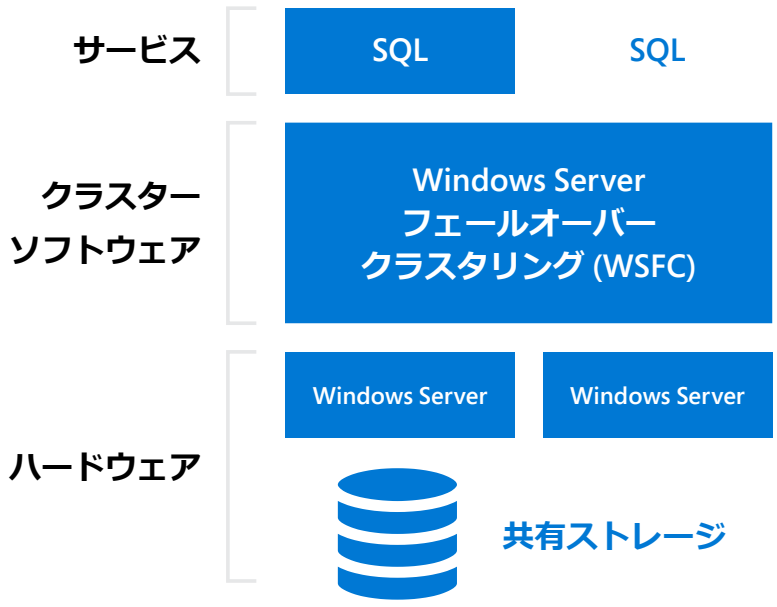


FCI では、クラスター マネージャーがクラスター リソースをマーシャリングする必要があります。SQL Server FCI は、Windows と Linux の両方で利用できます。Windows では、FCI は Windows Server フェールオーバー クラスタリング (WSFC) をクラスター マネージャーとして使用します。Linux では、Pacemaker というオープン ソースのクラスター マネージャーと Corosync というオープン ソースのサーバー通信システムを併用します。どちらのオペレーティング システムが使用されているとしても、これらのツールはほぼ同じ機能を実行し、アクティブ / パッシブ クラスター構成を作成します。

フェールオーバー クラスター インスタンス : Windows 上の SQL Server

Windows 上の FCI では、WSFC 機能を活用して、インスタンス レベルでの冗長性によるローカル高可用性を提供します。WSFC には、SQL Server の HADR シナリオをサポートするインフラ機能が備わっています。指定の可用性グループに属する可用性レプリカの現在のロールを監視および管理します。同時に、フェールオーバー イベントが可用性レプリカにどのような影響を及ぼすかを判別します。WSFC リソース グループは、作成する可用性グループごとに作成されます。WSFC はこのリソース グループを監視して、プライマリ レプリカの正常性を評価します。

FCI を使用すると、クラスター内に存在するマスター ノードまたはアクティブ ノードは常に 1 つになります。残りのノードは、セカンダリ モードまたはパッシブ モードになります。SQL Server の FCI は WSFC リソース グループ内で実行されます。リソース グループ内の各ノードが、構成設定とチェックポイントを使用したレジストリ キーの同期コピーを維持して、フェールオーバー後に FCI が完全に機能できるようにします。一度にクラスター内の 1 つのノードだけが、リソース グループを所有します。障害が発生した場合、リソース グループの所有権は別の WSFC ノードに移動します。このプロセスは、SQL Server に接続するクライアントまたはアプリケーションに対して透過的です。これにより、アプリケーションまたはクライアントが障害中に経験するダウンタイムを最小限に抑えることができます。



ドメインに参加しているサーバー

以前のバージョンの Windows Server では、WSFC 上に FCI を作成するには Active Directory が必要でした。Windows Server を使用して WSFC をデプロイするには、WSFC に参加するノードが同じ Active Directory ドメインに必ず参加していなければなりませんでした。WSFC が作成されると、Active Directory にクラスター名オブジェクト (CNO) またはアカウント、および対応する仮想コンピューター オブジェクト (VCO) が生成されます。しかし、WSFC と Active Directory 間のこの依存関係が、FCI のデプロイと管理においては大きな課題であり、障害となりました。この問題を解決するため、SQL Server ではドメインに依存しないフェールオーバー クラスターが導入され、管理者は Active Directory ドメインを使用せずに WSFC をデプロイできるようになりました。

WSFC ストレージ構成

FCI では、データベースとログ ストレージ用に FCI のすべてのノード間で共有ストレージを使用する必要があります。WSFC の各 FCI ノードには、標準の SQL Server フェールオーバー クラスター インスタンスのインストール環境ごとに共有ストレージが必要です。ストレージは、共有ディスク ストレージにファイバー チャネル、iSCSI、FCoE、または SAS を使用できます。または、記憶域スペース ダイレクト (S2D) を使用したローカルにアタッチしたストレージを使用することもできます。この共有ストレージを使用することにより、FCI 内のすべてのノードは、フェールオーバーがいつ生じてもインスタンス データに関して同じビューを持つことができます。共有ストレージは単一障害点となる可能性があります。FCI が確実にデータ保護を行うかどうかは、基礎となるストレージ ソリューションに依存します。

WSFC フェールオーバー

FCI を起動すると、いずれかのノードがリソース グループの所有権を持つものと想定し、そのノードの SQL Server インスタンスをオンラインにします。このノードが所有するリソースには、ネットワーク名、IP アドレス、共有ディスク、SQL Server データベース エンジン サービス、および SQL Server エージェント サービスがあります。任意の時点においては、リソース グループの個々の SQL Server インスタンスを実行しているリソース グループ所有者は 1 つだけで、FCI には他のノードは含まれません。フェールオーバーが発生すると、WSFC サービスはインスタンスのリソースの所有権を指定されたフェールオーバー ノードに転送します。その後、フェールオーバー ノードで SQL Server インスタンスが再起動され、データベースは通常どおりに回復されます。任意の時点で、FCI と基礎となるリソースをホストできるのはクラスター内の 1 つのノードのみです。

WSFC クォーラム層

WSFC 内の各ノードは、定期的なハートビート通信に参加して、ノードの正常性状態を他のノードと共有します。正常なノードは「アクティブ」とみなされ、応答しないノードは「障害あり」とみなされます。クォーラムとは、十分な応答リソースをオンラインにして WSFC の稼働を保証するメカニズムです。WSFC に十分な「投票」(アクティブ) メンバーが存在していればその WSFC は健全であり、ノード レベルのフォールト トレランスを提供することができます。

クォーラム モードにより、クォーラム投票に使用する方法や、自動フェールオーバーを実行するタイミング、またはクラスターをオフラインにするタイミングが決まります。計画外の障害、永続的なハードウェアの問題、または通信障害の結果として WSFC がオフラインになった場合は、手動介入が必要になります。管理者は、クォーラムを強制してから、フォールト トレランスを備えた構成で、残りのクラスター ノードをオンラインに戻す必要があります。

フェールオーバー クラスター インスタンス : Linux 上の SQL Server

Linux では、フェールオーバー クラスターは SQL Server と同じように機能しますが、構造は若干異なります。Linux 上の SQL Server は、単一の WSFC マネージャーの代わりに、WSFC 機能を実行するために連動する 2 つのコンポーネント、つまり Pacemaker と Corosync を使用します。

Linux はパッケージ ベースの OS で、システム サービスを個別にまたはグループに分けてインストールし、さまざまな機能を提供します。Red Hat Enterprise Linux (RHEL)、Ubuntu、 お よ び SUSE Linux Enterprise Server (SLES) の 3 つのサポート対象 Linux ディストリビューションには、Pacemaker と Corosync を組み込んだ高可用性機能が含まれています。Pacemaker は、操作の頭脳であり、クラスター メンバーの監視、クラスター リソースの管理、アクティブにするノードに関するポリシー ベースの決定を行います。Corosync は、Pacemaker とそれが管理するノードとの間の専用の双方向通信を提供します。

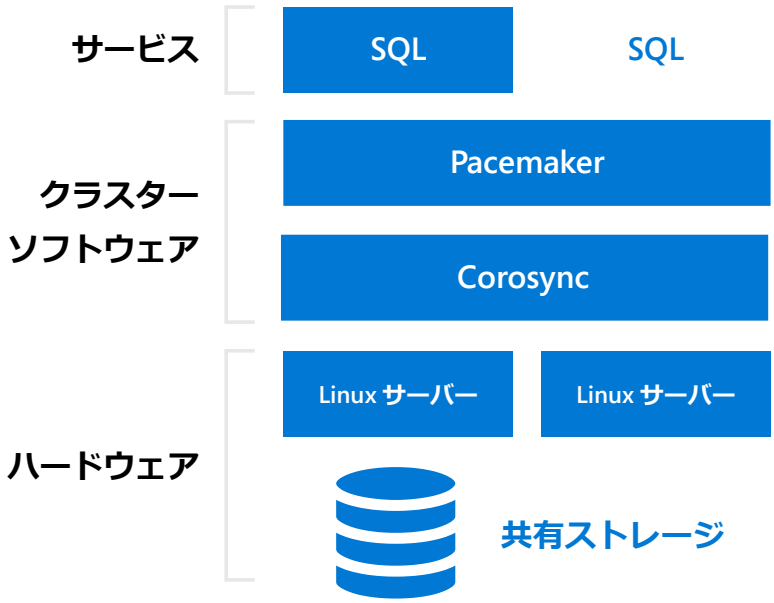
Linux のフェールオーバー クラスタリングは、Windows Server のフェールオーバー クラスタリングと同様、インスタンス レベルの継続性、自動および手動のフェールオーバー、アプリケーションやクライアントの透過的なフェールオーバー、数分または数秒でのリカバリーを提供します。

クラスター共有ストレージ オプション

すべての FCI には何らかの形式の共有ストレージが必要です。このストレージは FCI をホストできるすべてのサーバーに存在しますが、任意の時点で FCI 用にストレージを使用できるのは 1 つのサーバーだけです。Linux における共有ストレージに使用できる選択肢には、iSCSI、ネットワーク ファイル システム (NFS)、Common Internet File System (CIFS) があります。Linux 上の任意のストレージ リソースと同様、共有データ ストレージは FCI のクラスターのノードごとにマウントします。

Pacemaker フェールオーバー

Pacemaker 管理クラスターでは、WSFC 管理クラスター同様、1 つのノードがアクティブと見なされ、他のノードは非アクティブと見なされます。アクティブなノードだけが共有ストレージ リソースに接続するアクセス許可を持ちます。Pacemaker が障害を検出すると、アクティブなノードを正常な別のクラスター メンバーに切り替え、使用可能なクラスター メンバーのプールから障害が発生したノードを削除します。このプロセスはフェンスと呼ばれます (詳細は、後述のフェンスと STONITH で取り上げます)。



Pacemaker のクォーラム

クォーラムは、WSFC 同様に Pacemaker で使用される一般的な高可用性用語です。予想されるノードの半数以上がオンラインである場合、クラスターはクォーラムを持つと言われます。これは、クォーラム構成によって、オンラインのままクラスターに含めることができる障害ノードの数を判別できるため重要です。Linux では、通信層の Corosync がクォーラム ポーリングを管理し、状態の変化を Pacemaker に通知します。クォーラムではなくなっても、Pacemaker はクォーラムが復元されるまでクォーラムを無視し、1 つのノードのみでクラスター サービスを維持できます。つまり、ユーザーはサービスは引き続き利用できます。ただし、ノードが 1 つだけのクラスターは、本当の意味で高可用性ではありません。したがって、Pacemaker は引き続きノードをオンラインに戻し、クォーラムを回復しようとします。

フェンスと STONITH

クラスター リソース マネージャー、つまり Windows 上の WSFC または Linux 上の Pacemaker は、クラスター ノードの状態を監視し、発生した問題を処理します。状況によっては、誤動作しているノードや通信していないノードの隔離が必要または賢明な場合があります。この隔離機能は、高可用性の用語で「フェンス」と呼ばれます。基本的にフェンスは、クラスター内のリソースに対するアクセスを制御する方法です。リソース レベル (Web サーバーやデータベース サーバーなどの特定のサービスへのアクセスをブロックする目的)、またはノード レベル (クラスター ノードが「アクティブ」にならないようにする目的) のいずれかで使用できます。フェンスによって、正常に動作していないサービスまたはノードからクラスターを保護したり、正常に動作していてもメンテナンスのためにサービスまたはノードにオフラインのマークを付けたりすることができます。

クラスターから正常に動作していないノードを削除する操作 (ノード レベルのフェンス) は STONITH と呼ばれ、「Shoot the Other Node in the Head」の略ですぐにシャットダウンすることを意味します。サーバー ハードウェアと予算に応じて同じ操作を行う方法がいくつかありますが、それぞれの方法ではノードの電源を切断します。通常は電源を突然切ります。STONITH では、人間の介入なしにノードをオンラインに戻すことができないようにするため、クラスター リソース マネージャー (Windows の場合は WSFC、Linux の場合は Pacemaker) に修正しないよう指示します。

予測可能なフェールオーバー時間

SQL Server インスタンスが最後にチェックポイント操作を実行した時期に応じて、バッファ キャッシュにかなりの数のダーティ ページが存在する可能性があります。その結果、残りのダーティ ページをディスクに書き込むまで、フェールオーバーは続きます。これにより、フェールオーバー時間が長く、

予測できなくなる可能性があります。FCI は間接チェックポイントを使用して、バッファ キャッシュに保持されるダーティ ページの数を抑えることができます。これにより通常のワークロードで消費するリソースが増えますが、フェールオーバー時間の予測が行いやすくなり、構成も容易になります。組織内のサービス レベル契約で高可用性ソリューションの復旧時間目標 (RTO) が指定されている場合に非常に役立ちます。

Windows と Linux 上の SQL Server FCI の比較

Windows 上の FCI と Linux 上の FCI の主な違いは次のとおりです。

Linux でサポートされているのは、サーバーごとに 1 つの SQL Server インストールのみです。それで、すべての Linux FCI は既定で単一ノード インスタンスになります。Linux で複数のインスタンスが必要な場合、コンテナを使用することをお勧めします。Windows ではサーバーごとに複数の SQL Server インストールをサポートしています。インストールの数は、実行している Windows のバージョンに依存するので、ご使用の OS の制限を確認してください。

処理できる FCI の数は OS クラスターによって異なります。Windows では、WSFC あたり最大 25 の FCI がサポートされます。Pacemaker クラスターに含めることができるのは最大で 16 FCI です。各ノードが単一インスタンスであるためです。

Pacemaker には最大 16 個のノードをサポートする能力がありますが、**Standard Edition の SQL Server を使用して Linux で実装される FCI** がサポートできるのは、クラスターあたり最大 2 つのノードです。

コンテナにおけるフェールオーバー クラスター

クラスター化されたコンテナを使用するフェールオーバー クラスターは、クラウド環境の Linux 上で実行される高可用性を実現する別の方法です。コンテナが提供するデプロイメントは機敏です。つまり、コンテナに迅速かつ簡単にスピン アップし、不要になったときに再びシャットダウンできます。仮想マシンと比較すると、仮想マシンにはオペレーティング システム全体のオーバーヘッドが追加されますが、コンテナはオペレーティング システムの上で実行されるため、特定のアプリケーションやサービスとは分離できます。一般にコンテナは、イメージと呼ばれる構成スクリプト (コンテナの内容を定義するテンプレート) を使用します。障害が発生した場合、クラスター マネージャーはイメージを使用して新しいコンテナをデプロイできます。通常は数秒しかかかりません。

高可用性用の永続ストレージを備えた Azure Kubernetes Service (AKS) は、高可用性環境でコンテナを使用する一例です。SQL Server インスタンスが Kubernetes コンテナ (「ポッド」と呼ばれます) で失敗すると、Kubernetes は同じ永続ストレージに接続する新しいポッドに自動的に再作成します。ポッド構成によっては、迅速なリカバリー ソリューションになります。

レプリカを使用した SQL Server コンテナ イメージに基づくコンテナが含まれるデプロイメントについて、次の例で取り上げます。

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: mssql-deployment
spec:
  replicas: 1
  template:
    metadata:
      labels:
---
      containers:
      - name: mssql
        image: microsoft/mssql-server-linux
        ports:
        - containerPort: 1433
        env:
        - name: ACCEPT_EULA
          value: “Y”
        - name: SA_PASSWORD
---
```

次にデプロイメントを作成します。

```
kubectl apply -f <Path to sqldeployment.yaml file>
```

➔ フェールオーバー構成における Kubernetes のデプロイメントに関する**詳細をご覧ください**。

可用性グループは、プライマリ / セカンダリ (アクティブ / パッシブ) 構成でデータベース レベルのデータ保護を提供します。

SQL Server は、Windows OS WSFC サービスと、Always On 可用性グループと Always On FCI をサポートする機能を使用します。各クラスター メンバー (ノード) は、可用性グループでデータベースをホストしているかどうかに関係なく、クォーラムの判別に使用されます。

可用性グループに参加している各インスタンス (レプリカと呼ばれます) は、レプリカの使用方法によってスタンドアロンとしても、Always On FCI の一部としても構成できます。SQL Server 2017 は、可用性グループのために **Always On** と**読み取りスケール**という 2 つの異なるアーキテクチャをサポートしています。

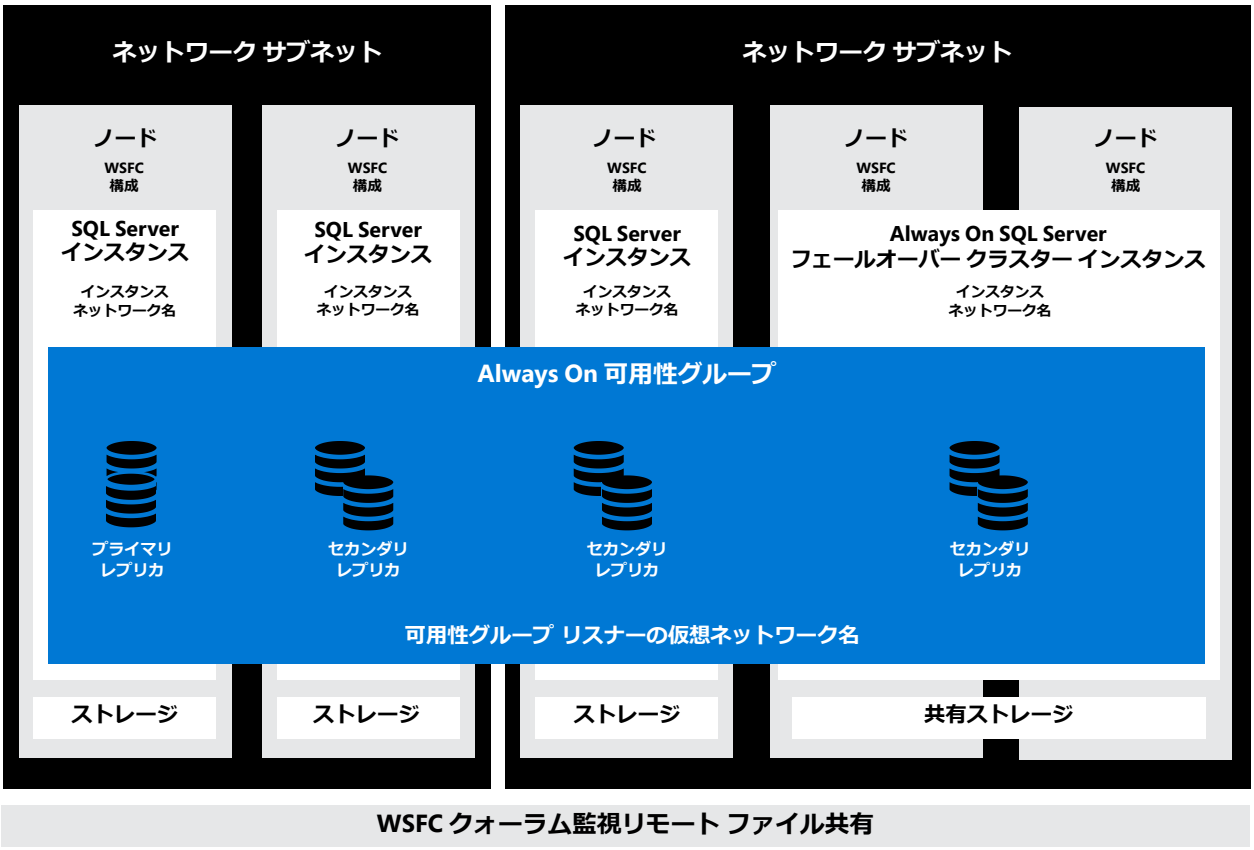
Always On 可用性グループ : この可用性グループは、データベースに高可用性とディザスター リカバリーを提供します。そのために、サーバー レベルのクラスター マネージャー (Windows 上の WSFC または Linux 上の Pacemaker のいずれか) を必要とします。クラスター マネージャーには、障害の検出、自動フェールオーバー、フェールオーバー後に高可用性を提供するための透過的再接続を行う機能が備わっています。また、各レプリカがそれぞれのクラスター ノードに存在していることも必要です。同期されたレプリカは、データベース レベルでタイムリーなディザスター リカバリーを行うためのデータ保護を提供します。

読み取りスケール可用性グループ : この可用性グループは、高可用性クラスターの一部となることなく、データ冗長性を提供できます。読み取りスケール可用性グループは、読み取り専用ワークロード用の他の SQL Server インスタンスにコピーされる、データベースのグループです。これらの機能は、Windows と Linux の両方の SQL Server で使用できます。

フェールオーバー クラスタリングを使用する可用性グループ

可用性グループは、フェールオーバー クラスターの高可用性を基盤として構築され、データベース レベルのクラスター ノード間に障害保護用の別のレイヤーを追加します。FCI は、SQL Server がデータ要求に確実に応答できるようにします。可用性グループは、戻されるデータが最も正確であるようにします。

また Linux では、フェールオーバー クラスタリングを備えた可用性グループもサポートしています。構成上の違いがいくつかあるものの、Linux 上の SQL Server の可用性グループは Windows 上の可用性グループと同じです。Windows で WSFC を使用するのと同じ場面で Linux では Pacemaker を使用してフェールオーバー クラスターを制御して正常性を監視します。



クラスターの種類

Windows には、[WSFC]、[外部]、[なし] という 3 つの種類のクラスターがあります。[WSFC] が Windows Server の既定の種類のクラスターで、この種類のクラスターのみがサーバー レベルの高可用性を提供します。ただし、ハイブリッド オペレーティング システムのクラスター環境が必要な場合は、クラスターの種類を [なし] に設定する必要があります。

Linux には、[外部] と [なし] の 2 つの種類のクラスターがあります。種類が [外部] のクラスターの場合、Pacemaker は対象の可用性グループ自体とその可用性グループ下の FCI を制御します。種類が [なし] のクラスターの場合、Pacemaker は FCI を制御している場合でも可用性グループを制御しません。種類が [なし] のクラスターが含まれる可用性グループでは、プライマリ レプリカからセカンダリ レプリカへのフェールオーバーは手動で行う必要があります。

可用性モデル

フェールオーバー クラスターにプライマリ インスタンスと対応するセカンダリ インスタンスがあるように、可用性グループにもプライマリ レプリカと対応するセカンダリ レプリカがあります。プライマリ レプリカはプライマリ データベースをホストし、読み取り / 書き込みトランザクションを許可します。プライマリ レプリカはすべてのトランザクションをログに記録し、各セカンダリ レプリカにそのログを送信して同期します。各セカンダリ レプリカはログのコピーを保存してからログに記録されたトランザクションをデータベースに適用します。このプロセスは「ハードニング」と呼ばれます。これは同期的に実行できます。つまり、プライマリ レプリカはセカンダリ レプリカが各トランザクション ログを保存するのを待機してから、別のレプリカに送信します。または非同期的、つまり、プライマリ レプリカはセカンダリ レプリカによるログの保存を待機せずに、別のレプリカに送信することもできます。

非同期コミット モードはプライマリ レプリカを高速化しますが、セカンダリ レプリカがプライマリに追いつこうとするため、セカンダリ レプリカのデータ整合性が遅れる原因となる可能性があります。同期コミット モードではプライマリ レプリカの速度が低下する可能性があります。プライマリはセカンダリ レプリカが追いつくのを待機しますが、セカンダリ データをプライマリ レプリカの可能な限り多くのデータと同期するために低下が生じます。

詳解 : 可用性モードの設定 - Windows および Linux

Windows または Linux 上の SQL Server に対して CREATE AVAILABILITY GROUP コマンドを使用すると、可用性モードを指定できます。

```
AVAILABILITY_MODE = { {SYNCHRONOUS_COMMIT | ASYNCHRONOUS_
COMMIT | CONFIGURATION_ONLY }
```

- SYNCHRONOUS_COMMIT は、プライマリ レプリカが、セカンダリ レプリカでトランザクションがハードニングされるまでコミットを待つよう指定します (同期コミット モード)。
- ASYNCHRONOUS_COMMIT は、セカンダリ レプリカがログをハードニングするのを待機することなく、プライマリ レプリカがトランザクションをコミットするよう指定します (同期コミット可用性モード)。
- SQL Server 2017 CU 1 では CONFIGURATION_ONLY が導入されました。CONFIGURATION_ONLY レプリカは、CLUSTER_TYPE = EXTERNAL または CLUSTER_TYPE = NONE が指定された可用性グループにのみ適用されます。CONFIGURATION_ONLY は、プライマリ レプリカが可用性グループ構成メタデータを対象レプリカ上のマスター データベースに同期的にコミットするよう指定します。これは、CLUSTER_TYPE = WSFC の場合には無効です。詳細については、マイクロソフトのドキュメントを参照してください。

➔ これらの可用性グループの**詳細については**、可用性モード (Always On 可用性グループ) を参照してください。

フェールオーバー モード

プライマリ レプリカが応答しなくなると、WSFC はフェールオーバーと呼ばれるプロセスでセカンダリ レプリカをプライマリ ロールに昇格させます。フェールオーバーを構成できる方法は、選択する可用性モードの種類 (同期または非同期) によって異なります。

可用性モデル	使用可能なフェールオーバー モード
同期コミット	計画的な手動フェールオーバーまたは自動フェールオーバー
非同期コミット	強制手動フェールオーバー

簡単に言えば、計画された手動フェールオーバーは、データベース管理者のコマンドによって開始されます。このコマンドにより、可用性クラスターではプライマリ レプリカのすべてのものがセカンダリ レプリカに同期され、同期されたセカンダリがプライマリ ポジションに昇格され、対象クラスターがセカンダリ レプリカになるよう指示が出されます。自動フェールオーバーはほぼ同じように移行しますが、プライマリ レプリカの障害に応答して発生します。自動フェールオーバーでは、応答可能な同期されたセカンダリ レプリカがプライマリ レプリカ ポジションに昇格されます。以前のプライマリ レプリカがオンラインに戻ると、セカンダリ レプリカとして指定されて同期されます。このセカンダリ レプリカは、新しいプライマリ レプリカと完全に同期するまでフェールオーバーには使用できません。

非同期コミット モードの場合にはトランザクション ログが正常に保存される (ハードニングされる) まで待機することなくプライマリ レプリカによってセカンダリにトランザクション ログを送信されます。フェールオーバー オプションとして利用できるのは強制手動フェールオーバーのみです。このフェールオーバーは、セカンダリ レプリカでハードニングされたトランザクション ログが対応するデータベースに適用されたかどうかに関係なく、すぐに開始されます。強制フェールオーバーにより、データの整合性が危険にさらされ、データが失われる可能性があります。

詳解 : Windows におけるフェールオーバー モード

SQL Server Management Studio (SSMS)、Transact-SQL (T-SQL)、または SQL Server PowerShell を使用して、可用性グループで定義された可用性レプリカのフェールオーバー モードを変更できます。SSMS では、[可用性レプリカのプロパティ] ダイアログ ボックスの [フェールオーバー モード] ドロップダウン リストを使用して、このレプリカのフェールオーバー モードを変更できます。

T-SQL では、`ALTER AVAILABILITY GROUP` ステートメントを使用してフェールオーバー モードを次のように変更します。

```
ALTER AVAILABILITY GROUP *group_name* MODIFY REPLICA ON
'*server_name*'
  WITH ( {
                | FAILOVER_MODE = { AUTOMATIC | MANUAL }
        } )
```

➔ **詳細については**、可用性レプリカの可用性モードの変更 (SQL Server) をご覧ください。

詳解 : Linux におけるフェールオーバー モード

Linux では、`CLUSTER_TYPE` が `EXTERNAL` の場合、外部クラスター マネージャーがすべての手動または自動のフェールオーバー処理を実行します。たとえば、ソリューションが Pacemaker を使用して Linux クラスターを管理する場合、`pcs` を使用して Red Hat Enterprise Linux (RHEL) または Ubuntu で手動フェールオーバーを実行します。

```
sudo pcs resource move ag_cluster-master nodeName2 --master
```

SUSE Linux Enterprise Server (SLES) では、`crm` を使用します。

```
crm resource migrate ag_cluster nodeName2
```

➔ Linux での Always On 可用性グループのフェールオーバーの**詳細については**、手動と強制のフェールオーバーに関するマイクロソフトのドキュメントをご覧ください。

可用性グループ リスナー

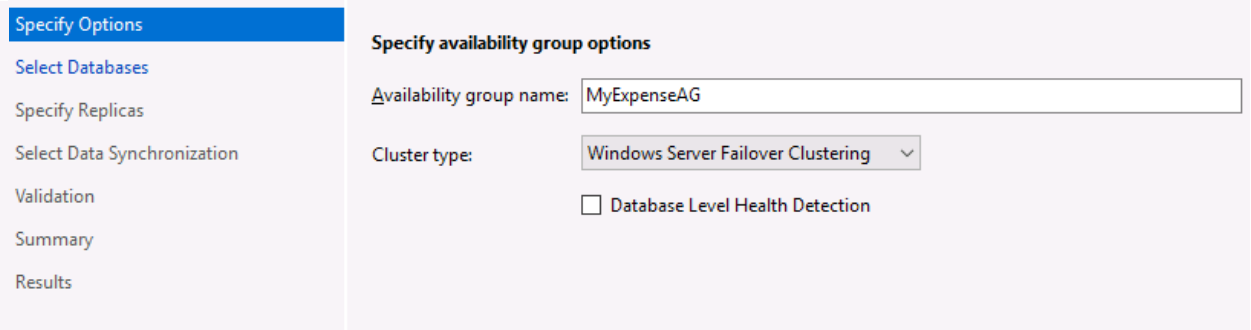
Always On 可用性グループはリスナー サービスを使用します。リスナー サービスを使用すると、アプリケーションとエンド ユーザーは、プライマリ レプリカをホストしている SQL Server インスタンスを把握する必要なく接続できます。Windows は、仮想 IP、仮想ネットワーク名、および DNS 構成を使用してこれを行います。各可用性グループには独自のリスナーがありますが、プライマリ レプリカのノードだけが、仮想ネットワーク名に接続する受信クライアント要求に応答します。

Windows の場合と同様、リスナーは Linux 上の可用性グループのオプション機能です。FCI のインスタンスと同じように可用性グループ リスナーは、インスタンスの名前を必要とせずに単一のクライアント接続ポイントを提供します。Linux では、任意のノード上で実行可能な、Pacemaker で作成された IP アドレス リソースがあります。「フレンドリ名」を持つ DNS 内のリスナーの IP リソースと関連付けられたエントリーが使用されます。リスナーの IP リソースは、対象の可用性グループのプライマリ レプリカをホストしているサーバー上でのみアクティブになります。

詳解 : Windows での可用性グループの設定

Windows で可用性グループを作成してセットアップするには、SQL Server Management Studio (SSMS)、Transact-SQL (T-SQL)、 お よ び SQL Server PowerShell の以下のいずれかのツールを使用できます。

1. **新しい可用性グループ ウィザード**。SSMS のこのウィザードを使用すると、可用性グループの作成と構成に必要なすべてのタスクを完了できます。たとえば、Windows の WSFC をクラスターの種類に指定できます。



➔ **詳細については**、可用性グループ ウィザードの使用 (SQL Server Management Studio) をご覧ください。

2. **T-SQL**。CREATE AVAILABILITY GROUP コマンドを使用して、以下のようにクラスターの種類として WSFC を指定します。

```
CREATE AVAILABILITY GROUP [ag1]
    WITH (DB_FAILOVER = ON, CLUSTER_TYPE = WSFC)
    FOR REPLICA ON
        N’agnode01’
        WITH (
            ENDPOINT_URL = N’tcp://agnode01:1433’,
            AVAILABILITY_MODE = SYNCHRONOUS_COMMIT,
            FAILOVER_MODE = EXTERNAL,
            SEEDING_MODE = AUTOMATIC
            ),
        N’agnode02’
        WITH (
            ENDPOINT_URL = N’tcp://agnode02:1433’,
            AVAILABILITY_MODE = SYNCHRONOUS_COMMIT,
            FAILOVER_MODE = EXTERNAL,
            SEEDING_MODE = AUTOMATIC
            );
```

➔ CREATE AVAILABILITY GROUP **構文の引数について詳しくは**、マイクロソフトのドキュメントをご覧ください。

3. **PowerShell**。PowerShell コマンドレットを使用して、SQL Server 2017 で Always On 可用性グループを作成および構成します。

```
# Create the availability group
New-SqlAvailabilityGroup `
    -Name “MyAG” `
    -Path “SQLSERVER:\SQL\PrimaryComputer\Instance” `
    -AvailabilityReplica @($primaryReplica,$secondaryReplica) `
    -Database “MyDatabase”
```

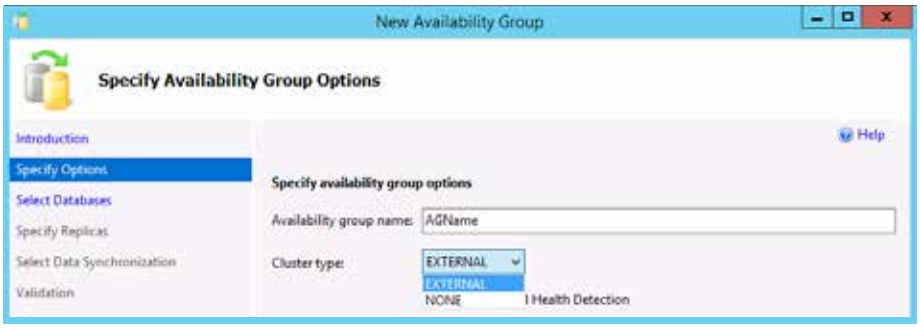
➔ **詳細については**、可用性グループの作成 (SQL Server PowerShell) をご覧ください。

詳解 : Linux での可用性グループの設定

高可用性のために Linux サーバー上に可用性グループを作成する手順は、Windows Server フェールオーバー クラスターでの方法とは異なります。Linux FCI では、エンドポイントと証明書を使用して Linux ノード上で可用性グループを有効にする必要があります。次に、Transact-SQL を使用して、または SSMS の新しい可用性グループ ウィザードを使用して、クラスターの種類が外部の可用性グループを作成できます。Pacemaker は外部クラスター エンティティの例です。

```
 CREATE AVAILABILITY GROUP [ag1]
WITH (DB_FAILOVER = ON, CLUSTER_TYPE = EXTERNAL)
FOR REPLICA ON

        ),
```



SQL Server で可用性グループを作成した後、クラスターの種類を外部に指定した場合には対応するリソースを Pacemaker で作成する必要があります。可用性グループに関連するリソースには、グループ自体と IP アドレスの 2 つがあります。

```
sudo pcs resource create ag_cluster ocf:mssql:ag ag_name=ag
--master meta notify=true
```

IP アドレスと可用性グループが確実に同じノードで実行されるようにするには、コロケーション制約を設定する必要があります。IP アドレスの前に可用性グループが稼働していることを確認するため、順序付けの制約を作成できます。この方法で、Linux 上の SQL Server の可用性グループを作成および構成できます。

➔ **ステップバイステップのガイダンスについては**、Linux 上の SQL Server の可用性グループの作成と構成に関するマイクロソフトのドキュメントをご覧ください。

Windows ベースと Linux ベースのハイブリッド可用性グループ

可用性グループは、複数のオペレーティング システムにまたがることができます。つまり、ハイブリッド可用性グループ内のレプリカを異なるオペレーティング システムで実行できます。ただし、Windows と Linux はレプリカ フェールオーバーの管理に異なるコントローラー (WSFC と Pacemaker) を使用するため、異なるオペレーティング システムが含まれる可用性グループではクラスターの種類 [なし] を使用し、手動でフェールオーバーを行う必要があります。

詳解 : Windows ベースと Linux ベースのハイブリッド可用性グループ

クロスプラットフォーム構成でハイブリッド可用性グループを作成できます。たとえば、Windows Server がプライマリ レプリカをホストし、Linux サーバーがセカンダリ レプリカをホストする可用性グループです。ただし、この可用性グループを作成する際は、クラスター マネージャーがないため、クラスターの種類 [なし] を選択する必要があります。

```
CREATE AVAILABILITY GROUP [ag1]
    WITH (CLUSTER_TYPE = NONE)
    FOR REPLICA ON
        N'<WinSQLInstance>'
     WITH (
        ENDPOINT_URL = N'tcp://<WinSQLInstance>:5022',
        AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
         SEEDING_MODE = AUTOMATIC,
         FAILOVER_MODE = MANUAL,
        SECONDARY_ROLE (ALLOW_CONNECTIONS = ALL)
         ),
        N'<LinuxSQLInstance>'
    WITH (
         ENDPOINT_URL = N'tcp://<LinuxSQLInstance>:5022',
        AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
        SEEDING_MODE = AUTOMATIC,
        FAILOVER_MODE = MANUAL,
        SECONDARY_ROLE (ALLOW_CONNECTIONS = ALL)
        )
GO
```

セカンダリ レプリカを可用性グループに参加させるには、クラスターの種類として外部を指定しなければなりません。

```
ALTER AVAILABILITY GROUP [ag1] JOIN WITH (CLUSTER_TYPE = NONE)
ALTER AVAILABILITY GROUP [ag1] GRANT CREATE ANY DATABASE
GO
```

プライマリ レプリカでは読み取りと書き込みが許可されます。プライマリであるレプリカを変更するには、フェールオーバーできます。高可用性の可用性グループでは、クラスター マネージャーがフェールオーバー プロセスを自動化します。クラスターの種類が [なし] の可用性グループの場合、フェールオーバー プロセスは手動です。■

フェールオーバー クラスタリングは、高可用性の一部に過ぎません。可用性のプライマリの場所で地震や洪水などの壊滅的な出来事が生じる場合、

ビジネスのシステムを他の場所でオンラインにするよう準備する必要があります。Always On FCI、ログ配布、Always On 可用性グループなどの機能が、必要なビジネス継続性とディザスター リカバリーを支援します。

Always On FCI

ディザスター リカバリーにおける FCI に関しては、他の考慮事項があります。共有ストレージです。プライマリ サイトとセカンダリ サイトでは同じディスクが使用できなければなりません。この可用性を確保するには、ハードウェア層でストレージ ベンダーが提供する機能などの外部の方法が必要になります。また、Windows の記憶域レプリカや、Linux における Unison などのストレージ同期を使用して、FCI が使用するディスクが他の場所に確実に存在するようにできます。

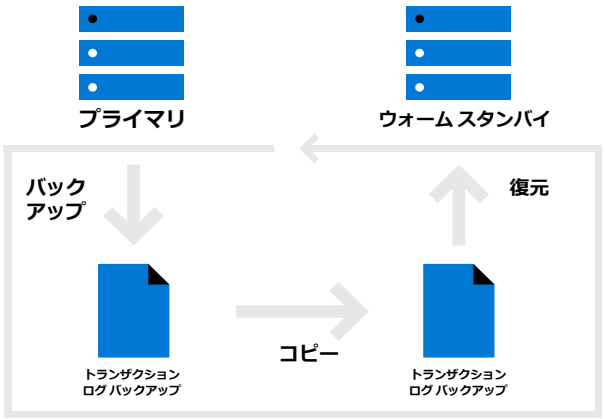
Always On 可用性グループ

Always On 可用性グループの利点の 1 つは、高可用性とディザスター リカバリーの両方を 1 つの機能を使用して構成できることです。共有ストレージも高可用にする必要はありません。高可用性を確保するため 1 つのデータセンターにローカルなレプリカを持ち、ディザスター リカバリーのために別のデータセンターにリモート レプリカを持ち、それぞれに別個のストレージを備えるほうが簡単です。データベースの追加コピーを持つという手段は、冗長性を確保するためのトレードオフとなります。

ログ配布

ログ配布は、SQL Server に備わっている高可用性をサポートする機能です。ログ配布を使用すると、プライマリ サーバーにあるデータベースのトランザクション ログがバックアップされ、1 つ以上のセカンダリ データベースに送信され、それらのセカンダリ データベースで復元されます (図を参照)。これにより、セカンダリ データベースはプライマリと同期した状態に保たれます。このプロセスは、事前構成された間隔で自動的に行われ、トランザクション ロールバックが可能になります。同期の間隔が長くなればなるほど、フェールオーバーの時間が長くなります。

ログ配布は、Windows と Linux のどちらでも利用できます。Linux では、SQL Server エージェント ジョブは SQL Server の基本インストールには含まれていません。ログ配布を使用するためにインストールする必要がある mssql-server-agent パッケージと一緒に追加されます。Linux 上の SQL Server は Common Internet File System (CIFS) と Samba ネットワーキング プロトコル サービスを使用して、トランザクション ログ バックアップをネットワーク ファイル共有に格納します。Linux 上の SQL Server エージェントは、ログ バックアップをセカンダリ サーバーに転送するストアド プロシージャを定期的に実行します。Windows の場合と同様、ログ配布を使用して保護されているデータベースを復元する作業は手動です。■



重要なビジネス アプリケーションのダウンタイム、データの運用停止、アプリケーション パフォーマンスを維持する必要性は、今日のビジネスにとって最上位の関心事となっています。

ダウンタイムが発生してもビジネス継続性を確保するには、組織が高可用性 (HA) とディザスター リカバリー (DR) の戦略を検討することが不可欠です。Microsoft Windows Server および Microsoft SQL Server に備わっている HA と DR のソリューションを使用すると、可用性グループの多様な機能セット、フェールオーバー クラスター インスタンス、ログ配布の機能によって、ビジネスを常に有効で可用性のある状態に保つことができます。また可用性グループは、同じアーキテクチャの一部としてデータベースの追加コピーを提供することにより、読み取り可能なコピーをスケールアウトすることもできます。SQL Server のインスタンスとデータベースは、同じ機能を使用して Windows と Linux の両方で高可用性を実現できるようになりました。■

SQL Server の高可用性ソリューションの詳細。

SQL Server の Always On フェールオーバー クラスター インスタンスについて参照する。

SQL Server の Always On 可用性グループの詳しいインサイトを得る。