EDBホワイトペーパー

Postgresのデータベースをコンテナにデプロイする

入門

EDB™ POSTGRES

CONTENTS

2019年4月公開

EDB POSTGRES

はじめに

今や、コンテナとコンテナオーケストレーションがハイプの段階を越えて、主流のアプリケーション開発技術に急速になりつつある、と言ってもよいでしょう。コンテナの成長トレンドを見れば説得力があります。

- ClearPath Strategies が実施した600人以上のIT意思決定者を対象とした調査によ れば、企業の30%が既にコンテナをデプロイしており、42%は評価段階にあります。
- Sysdigのレポートによれば、1サーバー当りのコンテナ数の中央値は15程であり、前年比で50%増加しています。
- コンテナは、開発/テスト環境で実験されているだけではありません。500人以上のITプロフェッショナルを対象としたPortworxの調査で、2017年の67%に対して、2018年にはコンテナを実行している企業の83%が本番環境でコンテナを使用していたことがわかりました。
- Diamantiのコンテナ採用ベンチマーク調査では、新しいクラウドネイティブアプリケーション（55%）、軽量のステートレスアプリ（39%）、レガシーアプリの最新 化（31%）、データベース（30%）など、コンテナのユースケースが劇的に多様 化していることが明確に示されています。
- 最後に、451のリサーチが、2018年におけるコンテナテクノロジーからの収益は15億3,000万ドルで、2015年の3倍強であったと報告しています。2020年までにコンテナテクノロジーの収益はさらに75%上昇し、27億ドルに達すると予測されています。

ところで、コンテナとはなんでしょうか？コンテナとは、環境（オンプレミス、パブリックまたはプライベートクラウド、ベアメタル）に依存せずにLinuxまたはWindowsプラットフォームでアプリケーションを起動し実行するために必要なすべてのコード、構成、依存関係をパッケージ化した、リソースから分離されたソフトウェアオブジェクトです。複数のコンテナで1つのオペレーティングシステム（OS）を共有できるため、仮想マシン（VM）よりもずっと軽量になります。Kubernetesなどのコンテナ オーケストレーション プラットフォームでは、コンテナのクラスタを自動的に起動し、ライフサイクル全体を管理できるため、大規模なコンテナのデプロイメントを簡単に行うことができます。

このホワイトペーパーでは、コンテナの成長を推進しているものが何か、そしてコンテナのユースケースの1つであるデータベースの可能性と課題について、ある程度詳しく説明します。

何がコンテナを促進しているのか？

Stephen O'Gradyは The Road to Abstraction という優れた記事で、1960年代 にCOBOLで始まった一連の仮想化にコンテ ナを配置し、プログラム開発者の機械語の ビットとバイトの負担を軽減しました。仮想化の歴史におけるもう1つの重要な節目が、Javaアプリケーションサーバーでした。 これは、20世紀から21世紀の変わり目に登 場し、Javaアプリと基盤となるOS /ハード ウェア インフラストラクチャの間に抽象化 レイヤーを提供しました。

同時にわたしたちは、コモディティサーバー上のインフラストラクチャのデプロイメントを抽象化、標準化する機能を備えたVMの台頭を目の当たりにしました。VMは依然として世界中のデータセンターに広くデプロイされています。今後も長年にわたって使用される可能性が高いでしょう。ですが、VMと並行して、あるいはVMの代替としてコンテナの利用を推奨し、そして、推奨し続ける理由は数多くあります。以下のような要因でコンテナの利用が推進されます。

- 増え続けるクラウド生まれのアプリとコンテナは、マイクロサービスやサーバーレスと同様に、クラウドネイティブ テクノロジーへのトレンドの重要な要素となっています。
- OSとインフラストラクチャレイヤーからアプリを完全に抽象化することで、コンテナは比類なき俊敏性と柔軟性を提供し、継続的な統合/配信/デプロイメントのDevOpsアプローチをサポートします。また、コンテナイメージはVMよりもはるかに速く起動するため、必要に応じてアプリがスケールアップ、ダウンすることが予想される今日の動的なランタイム環境により適しています。

- コンテナは、開発/テストから本番まで、アプリケーションのライフサイクル全体でコストを節約します。定量化された節約の事例は以下の通りです（Forresterのレポートに基づく）。
  - ハイパーバイザーのライセンス料が削減し、本番環境におけるコストが最大50%削減します。
  - 開発/テストコストを最大70%合理化するだけでなく、市場投入までの時間を短縮するプロセス最適化によって、収益などのビジネス価値をより迅速に獲得できるようになります。
  - OSのコピーを何度 も実行する必要がないため、VMデプロ イメントと比較して実行時のリソース消 費が節約されます。サーバーは最大80% 削減されます。
  - 独自のVMプラットフォームとは対照的に、コンテナは主にオープンソースです。コンテナを使用するエンタープライズ開発チームは、活気のあるオープンソースコミュニティを活用できます。また、ベンダーのロックインも回避されています。今日のコンテナの83%はロイヤリティフリーのDockerエンジンで実行されています。
  - 開発者とオペレーターの観点から見た場合、コンテナイメージ は、デプロイ先に関係なく、開発、ステージ ング、運用段階でまったく同じように実行さ れます。この一貫性により、環境の不一致が 原因で問題が発生したときによく耳にする「 私のマシンでは動作します」という返事がなくなります。

上記のメリットはすべて、コンテナの急速な普及に間違いなく貢献しています。ですが、コンテナの最も重要な命題は箇条書きの最後で暗に言及さ れている ア プリケーションのポータビリティです。ますます多様化、複雑化するハイブリッドおよびマルチクラウド環境が標準になっており、同じソフトウェアを複数のパブリッククラウドと プライベートクラウドで、またオンプレミスの仮想化環境でも同じ方法で運用できることは大きなメリットです。 アプリのデプロイメントとインフラストラクチャの要件はすべて、コンテナイ メージにパッケージ化されており、期待 どおりに実行されることを確信したうえ で、どこからでも起動できます。

また、オンデマンドで、クラスター全体でコンテナを管理、デプロイするコンテナオーケストレーション フレームワークがなければ、コンテナが主流とならなかったことも理解する必要があります。現在メジャーなフレームワークはKubernetesです。Kubernetesは、コンテナ管理のオープンソースのパイオニアであり、元はGoogleが開発しました。Kubernetesは、次のようなフレームワークの基盤でもあります。

OpenShift、GKE、Amazon EKS、IBM Cloud Privateは、Docker EnterpriseやCloud Foundry with PKSなど、元々Kubernetesベースではなかったフレームワークに組み込まれています。

オーケストレーションを使用すれば、アプリケーションとサービスを中断することなく、コンテナを自動的に再構成、スケーリング、アップグレード、更新、移行できます。オーケストレーションフレームワークは、ランタイムのコンテナのデプロイメントの正常性を監視し、フェイルオーバーや災害時における高可用性と継続性を保証します。

また、負荷を監 視し、需要の変化に応じてサービスを自 動的に拡大、縮小できます。コンテナオーケストレータは、コンテナ間、あるいは外部環境へのネットワーク化を促進します。ローカルまたはクラウド内の、接続された永続または一時ボリュームの管理を通じて、通常はデータストレージを、とりわけステートフル性を処理します。

なぜ今、コンテナ化されたデータベースなのか？

コンテナ化されたデータベースはカプセル化されたDBMSサーバーソフトウェアであり、ネットワーク内のどこかにある物理データベースファイルにアクセスできるようになっています。各DBMSはそれぞれのコンテナイメージに収められています。しかし、データベースのコンテナ化は、アプリケーションのコンテナ化ほど簡単ではありません。よく言及されるデータベースのコンテナ化の課題には、以下のようなものがあります。

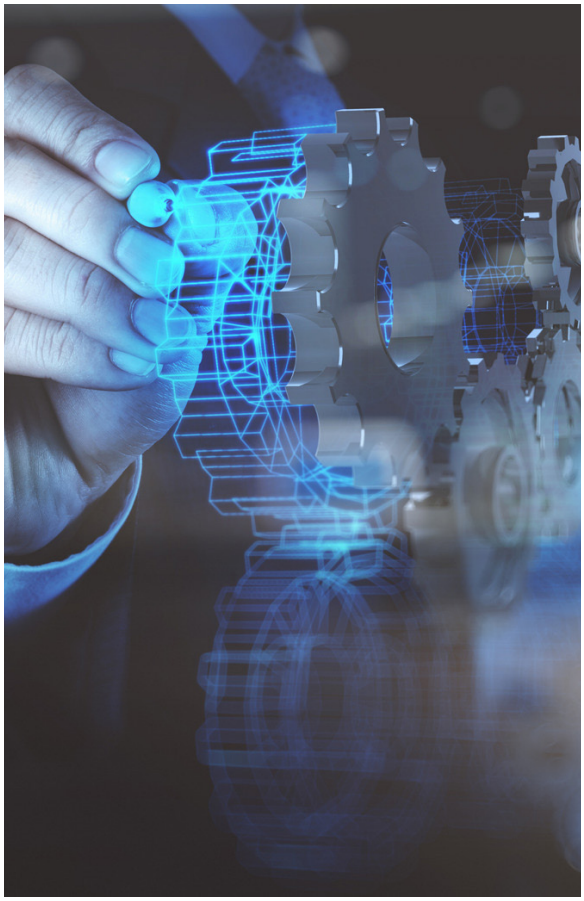
- 通常、データベースには高スループットで低遅延のネットワークが求められます。しかしDockerコンテナは、これらの要件を達成するために必要となるレベルのストレージとネットワークリソースの分離をネイティブに提供しません。
- コンテナ化されたデータベースに大量のデータを保存するためのディスク容量が必要となるため、機敏性が低下し、再配置が困難となります。
- データベースは元々ステートフルで耐久性がありますが、コンテナは通常、ステートレスで一時的なものです。永続的データストレージと通常よりも長いコンテナのライフスパンを処理するための回避策により、ランタイムリソースの使用量が削減されるというコンテナの大きなメリットがしばしば損なわれます。
- 通常のデータベースには多数の調整パラメーターがあり、その多くが動的なものです。考えうるすべてのデータベース構成に対して新しく不変のコンテナイメージを構築すると、イメージが無秩序に拡大することになります。また、コンテナはVMよりもかなり軽量です。そのため、この問題はVMのデプロイメントではさらに困難になることに注意してください。

コンテナ化されたデータベースの代替として、サービスプロバイダーがパフォーマンス コンフィグレーションを含む、必要となるデータベース物理インフラストラクチャとサーバーサイドDBMSリソースを担当するAPIベースのクラウドサービスモデルであるDatabase-as-a-Service（DBaaS）があります。

結論

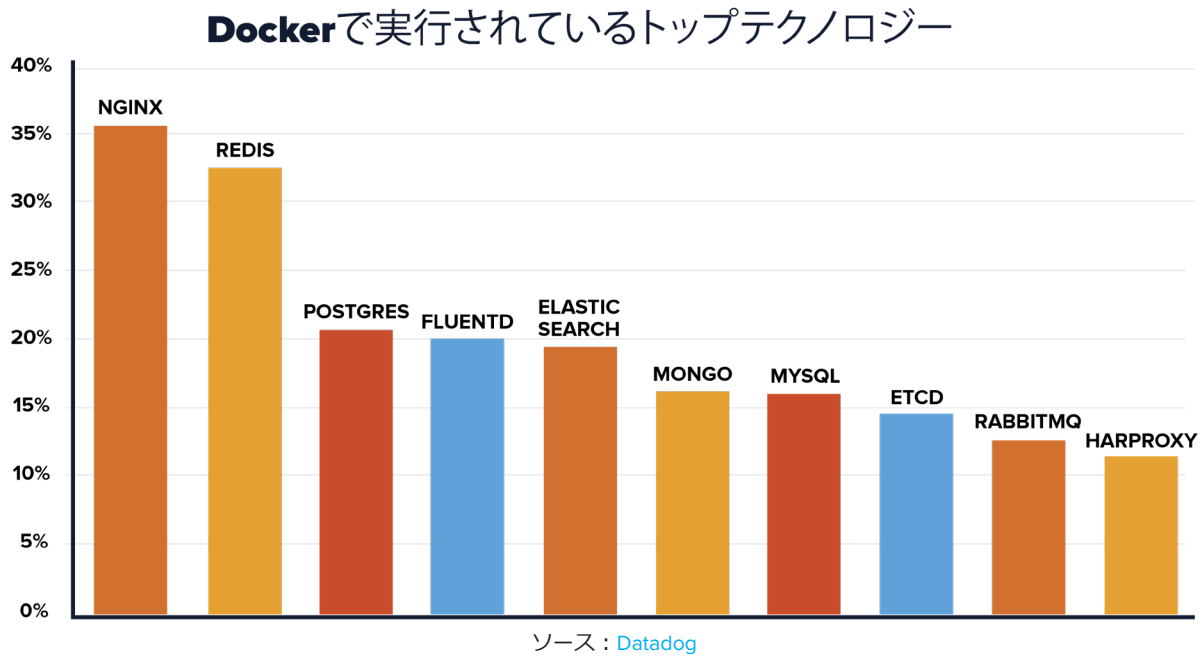
最終的に、各プロジェクトチームは、コンテナ化されたデータベースとDBaaSのどちらが、全体的なアプリケーション アーキテクチャとその開発プロセスに最適なクラウドデータベース プロビジョニングであるかについて、独自に決定を下す必要があります。DBaaSクラウドデプロイメントの主なメリットは生産性と管理の簡略化であり、コンテナ化されたデータベースの場合はポータビリティと自動化です。いずれにせよ、クラウド内のデータベースへの最適なルートを選択するときは、以下に示すコンテナ化されたデータベースの特性を考慮する必要があります。

- 今では大規模なモノリシック アプリケーションよりも好まれているマイクロサービス アーキテクチャには、小さくコンテナ化されたデータベースが適しています。データベースが複数のアプリケーションの中心的なリソースとなっているDBaaSデータベースのデプロイメントとは対照的に、コンテナ化されたデータベースは、本質的に、サービスを提供する特定のアプリケーションのコンポーネントとなっています。
- コンテナ化されたデータベースの自動化/スクリプト化されたデプロイメントをオーケストレーション フレームワークと組み合わせて使用すると、従来のクラスタリングと連携して、ステートフル システムとステートレス システムの両方の高可用性を実現できます。



- コンテナ化されたデータベース固有の弾力性により、柔軟性が高く無駄のないデータベース容量の先行計画とプロビジョニングが可能になります。コンテナを使用すると、オンデマンドユーティリティとしてデータベースにアクセスできるようになります。
- 同様に、コンテナ化されたデータベースではストレージとコンピューティングが分離されるため、ストレージのパフォーマンスと容量を、コンピューティングリソースとは無関係にスケーリングさせることができます。
- ソフトウェア定義のコンテナ化されたデータベースは、スピードが速いDevOpsサイクルで重要となるミッシングリンクを提供し、開発チームと運用チームがシームレスに連携できるようにします。

コンテナ化されたデータベースがトレンドになっていることを示唆するものとして、よく知られているオープンソースのリレーショナルデータベースであるPostgresが、現在Dockerで実行されている3番目に人気のあるテクノロジーであるという事実があります.

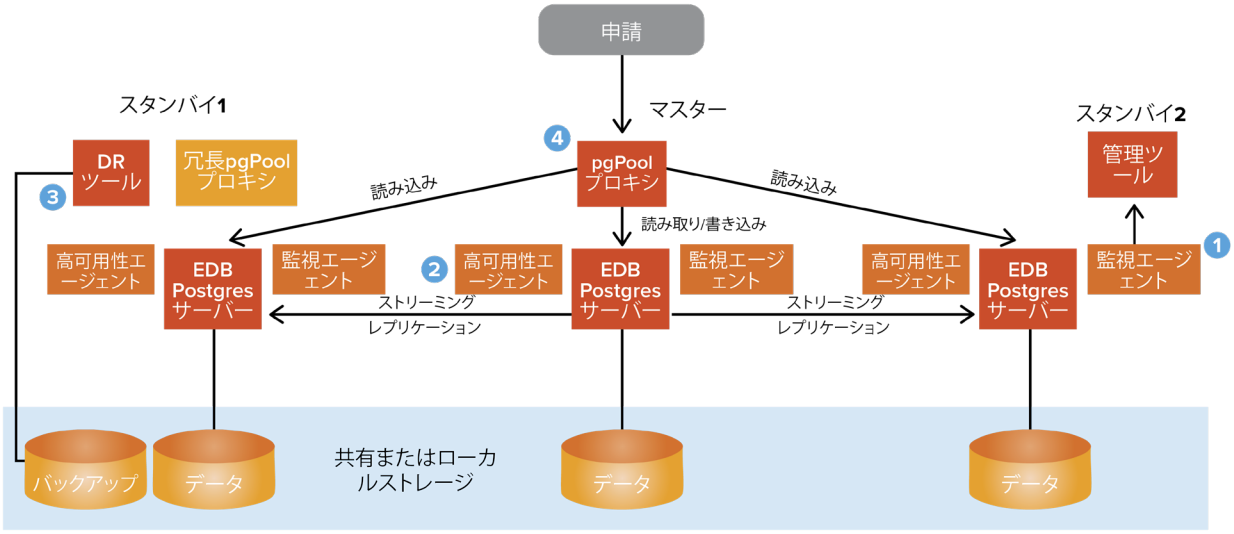


ソース：Datadog

EDB Postgresがコンテナ化をどのように加速するか

コンテナ化されたPostgresのデプロイメントのアーキテクチャをレイアウトする前に、以下に示す、高可用性用に構成された標準Postgres環境がどのようなものかを理解することが重要です。

代表的なデプロイメント



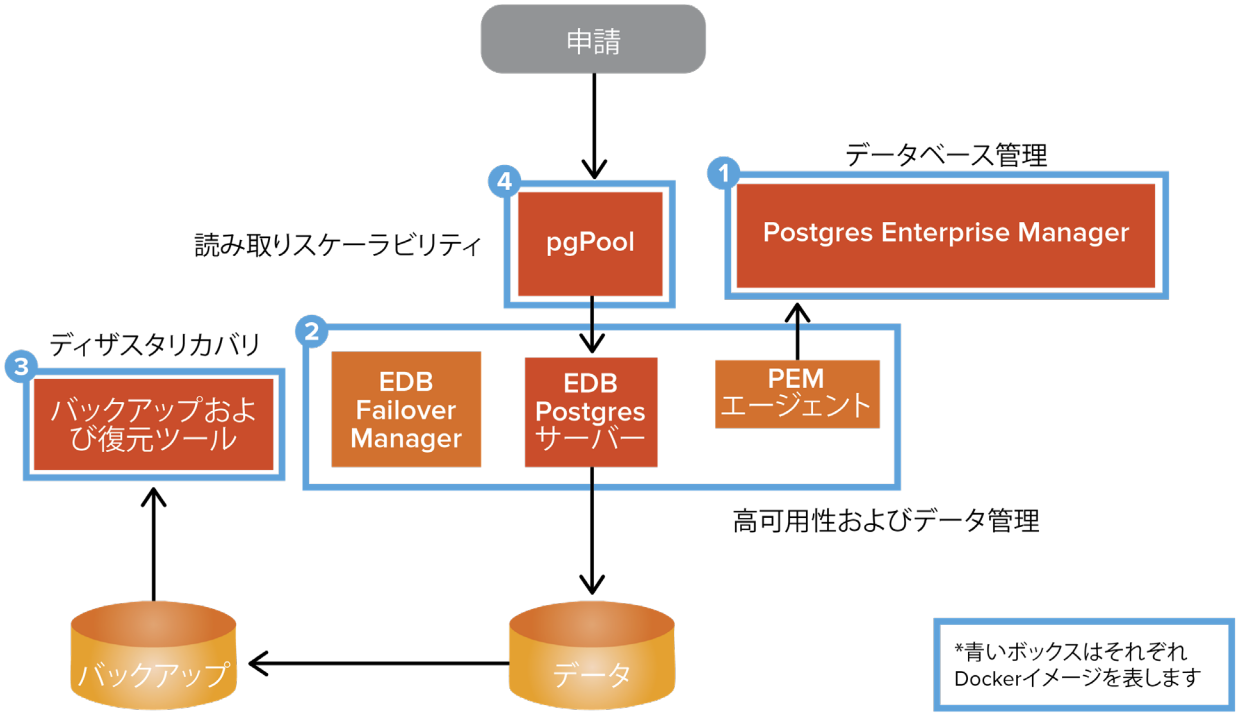
1. 監視と管理：それぞれのPostgresサーバーがエージェントによって監視され、トラッキングされたデータが管理ツールに報告されます。
2. 高可用性： EDB Postgresが、増分変更をマスターから任意の数の読み取り専用のスタンバイレプリカにストリーミングします。高可用性エージェントが障害を監視し、検出された障害の性質に応じたフェイルオーバープロトコルを自動的にトリガーします。
3. ディザスタリカバリ：EDB Postgresがバックアップデータストアを保持し、必要に応じてエンタープライズレベルのディザスタリカバリ手順を自動調整します。
4. スケーラビリティ：EDBはpgPoolプロキシのメジャーコントリビュータのひとりです。読み取りトランザクションをレプリカにロードバランシングし、マスターに書き込みのリクエストを送信することにより、アプリケーションのスケーリングをサポートします。また、フェイルオーバーの状況において、高可用性エージェントによってアプリケーションが自動的に接続される冗長pgPoolプロキシもあります。

大規模に運用できる、復元力のあるPostgresアーキテクチャを作成するには、次の4つの機能を含める必要があります。

- 高可用性のためのFailover Manager
- データベースの監視と管理
- ディザスタリカバリのためのバックアップ
- スケーラビリティのためのクエリルーティングとロードバランシング

EDB Postgresは4つのコンテナとして提供され、そのすべてが**Kubernetes**により管理されます。

推奨されるコンテナ化されたデプロイメント



1. データベース管理このコンテナにパッケージ化されたPostgres EnterpriseManager（PEM）は、データベースを監視し、ダッシュボード内に表示される、アラート状態分析用のパフォーマンスおよびステータスデータを収集します。これらのアラートは、オペレータあるいは他のエンタープライズレベルの管理システムに中継できます。
2. 高可用性とデータ管理：このコンテナには、EDB Postgres Failover Manager（EFM）とEDB Postgres Advanced Serverが含まれます。ユーザーはKubernetesにマスターコンテナの複数のレプリカをスピンアップさせることができるため、マスターに障害が発生した場合に、レプリカの1つに自動的に引き継がせることができます。
3. ディザスタリカバリ：このコンテナには、複数の異なるコンテナ内のデータベースをバックアップできるEDB Postgres Backup and Recovery Tool （BART）が含まれています。これにより、BARTは複数のデプロイメントを監視できます。
4. リード スケーラビリティ：このコンテナは、pgPoolを使用してクエリルーティングと接続プーリングを提供します。このコンテナのメリットは、pgPoolがクエリをレプリカデータベースにルーティングするロードバランサとして機能する読み取りアクティビティをスケーリングできることです。他のコンテナの前にあり、データベースコンテナとは別にスケーリングできます。

コンテナの利用で誰が恩恵を受けられるか？

インフラストラクチャを最新化したいすべての
ユー ザーが恩恵を受けることができます。

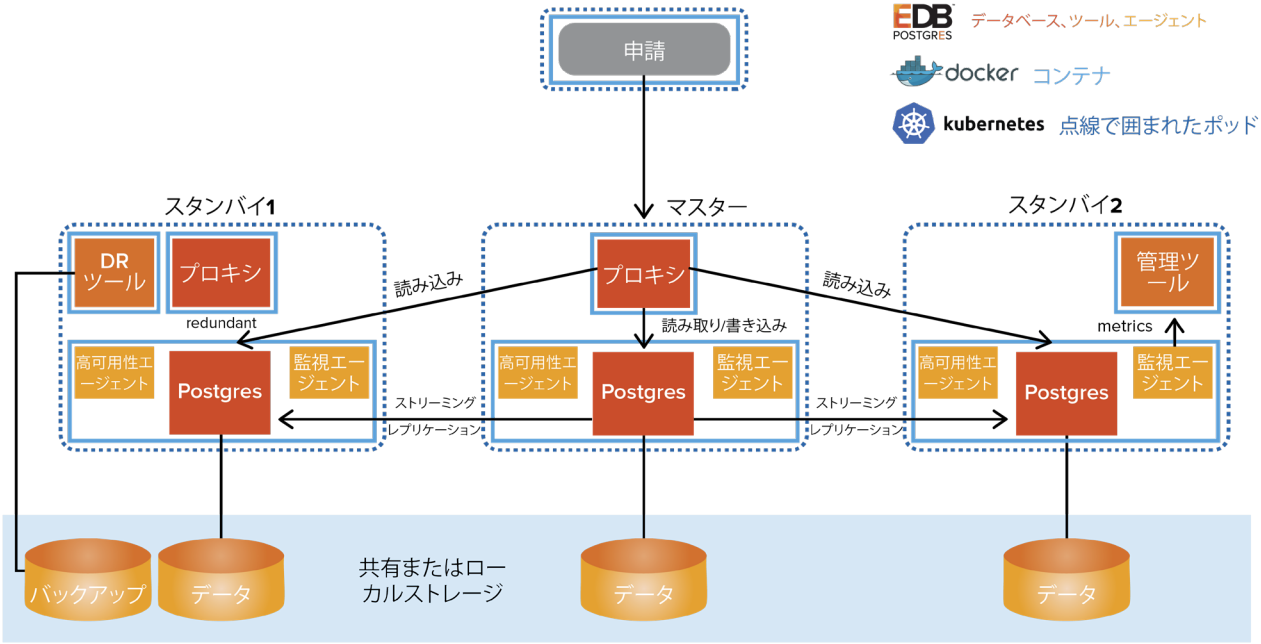
EDB Postgresはプラットフォームに依存しません。Red Hat OpenShift, Google Kubernetes Engine、Pivotal Cloud Foundr, Docker, IBM Cloud、Amazon Web Services（AWS）などのKubernetesプラットフォームにデプロイできます。

以下の図は、高可用性とフェイルオーバーのためのマスターポッド、および2つのレプリカポッドを備えた、本番環境におけるEDB Postgresの典型的な構成を 示しています。

レプリカポッドでデプロイメントを管理して冗長なデータベースサーバーを個別の 物理ハードウェア上に配置します。

管理と監視はPEMを介して行われます。共有共有ストレージはデータベース自体、およびバックアップとリカバリのために、本番環境で推奨されるデータストレージの方法です。

推奨される**Kubernetes**デプロイメント



EDBのユーザーはEDB Postgresを使用すると、コンテナ化されたPostgresデータベースの恩恵をすぐに受けることができます。大抵の場合、単純なアプリケーションから始まり、時間とともにより複雑なシステムへと移行します。EDBはユーザーと密に連携し、次世代のマイクロサービスベースのアプリケーションでコンテナを効果的に活用するためのベストプラクティスを共有しています。

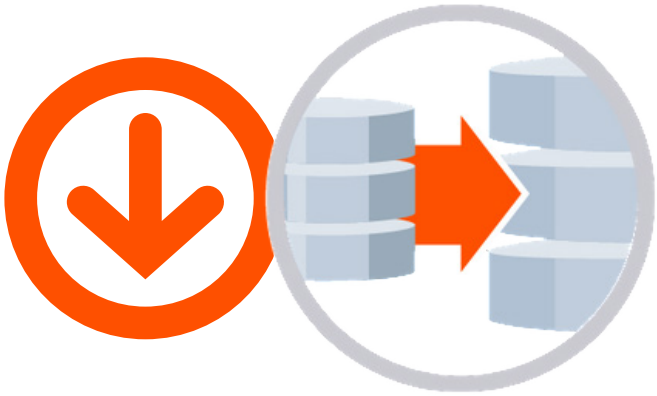
アーキテクチャー ロードマップとソリューション ブループリントサービスを通じて、EDBソリューションアーキテクトは、ユーザーが大規模で扱いにくい データベースに苦労している現在の状況から、戦略的なビジネス目標に完全に沿った、小さく機敏なデプロイメントという新しい世界に到達するための、カスタマイズされたコンテナロードマップの設計を支援します。EDBのユーザーは実績のあるブループリント、フルスタックツールセット、API統合設計のリファレンスライブラリを活用して、デジタ ルトランスフォーメーション イニシアチブを加速させ、ロバストなオープンソースベー スのデータアーキテクチャをモデル化し、複雑な環境に迅速にデプロイできます。

結論

つまり、コンテナとコンテナ オーケストレーションは、クラウドネイティブ イニシアチブの中心として位置付けられるまでに成熟しました。VMと同様に、コンテナは、コンピューティング/ストレージ/ネットワークインフラストラクチャレイヤーからアプリケーションレイヤーを抽象化します。またさらに一歩進んで、コンテナはオペレーティングシステムレイヤーも抽象化します。その結果、軽量で、リソースが最適化された、自己完結型のアプリケーションを、幅広い多様なプラットフォームで一貫して実行できるようになります。世界中の企業が、開発/テストおよび本番のワークロードでコンテナを使用して、クラウド生まれの製品とサービスを迅速に開発、デプロイし、簡単にスケーリングできるようになります。

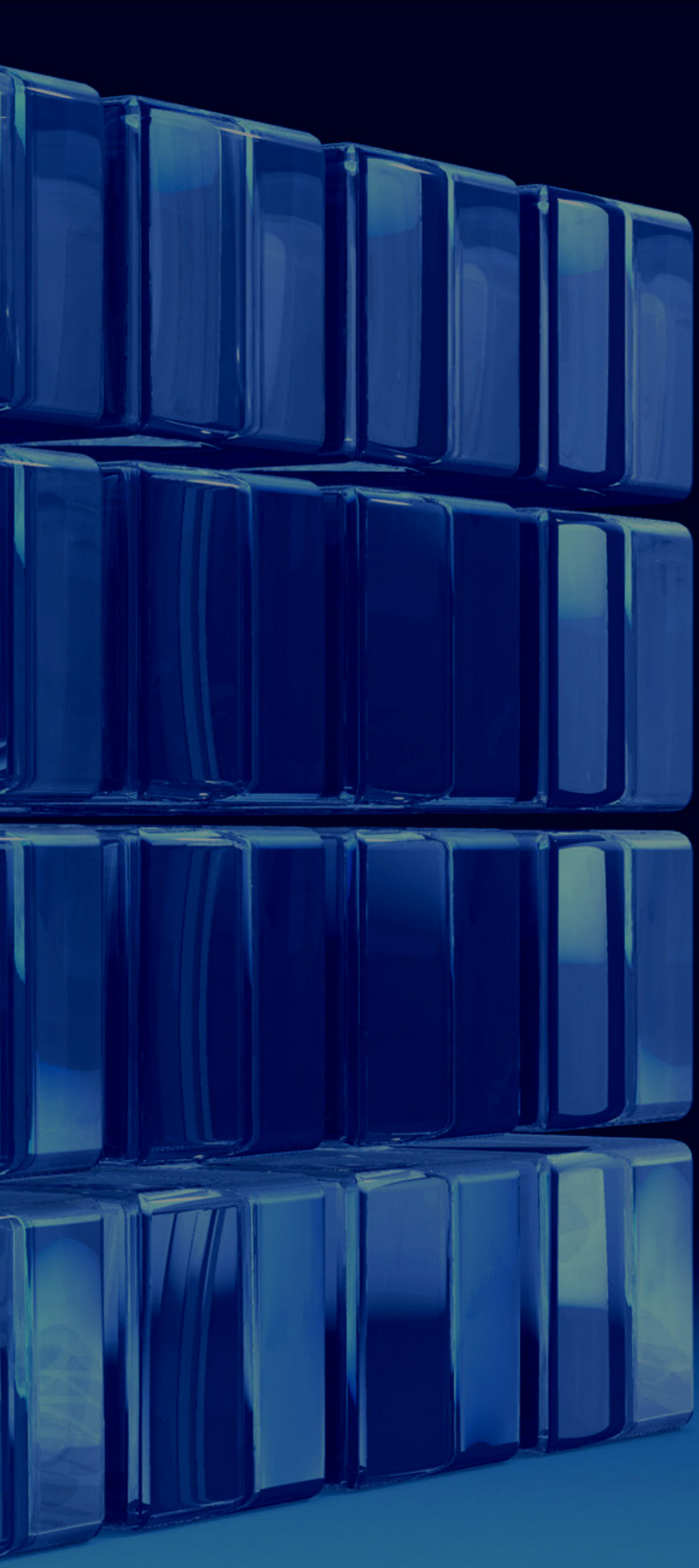
コンテナ化されたアプリケーションに加え、コンテナ化されたデータベースが、大規模なモノリシック アプリケーションからマイクロサービスで構成されるアプリケーションへのパラダイムシフトの一部として登場しました。コンテナ化されたデータベースは、複数のアプリケーションにサービスを提供する大規模集中型データベースではなく、アプリケーション自体に欠かせないオンデマンドユーティリティとなりました。とはいう ものの、とは言うものの、コンテナ化されたデータベースに関して、データの高可用性、バックアップとリカバリ、その他重要となるデータベースのパフォーマンスとコンプライアンスの要件など独自の課題が残っています。

EDB Postgresを使用することで企業は、重要となるデータベース管理と監視の要件を犠牲にすることなく、コンテナ化されたデータベースを活用できます。Kubernetesと連携したネイティブPostgres Dockerコンテナを用いて、EDB Postgresが高いデータ可用性、シームレスなフェイルオーバー、スケーラビリティとロードバランシング、自動化された監視とチューニング堅牢なバックアップとリカバリを保証します。



EDBのWebサイトから、EDBのコンテナをダウンロードしてお試しください（[リポジトリにアクセス]ボタンをクリックしてください）。サポートされているオプションとプラットフォームの詳細や、ドキュメントを見ることもできます。

www.enterprisedb.com/containers



EDB™
POSTGRES

EnterpriseDB | www.enterprisedb.com