Databricks 技術規範文書

1. はじめに

2. Databricks 概要

3. Databricks アーキテクチャ

4. Databricks ベストプラクティス

5. Databricks 設定ガイド

6. Databricks 使用ガイドライン

7. まとめ

Databricksは、データエンジニアリング、機械学習、データサイエンス、データウェアハウジングを統合したクラウドベースのデータおよびAIプラットフォームです。Apache Sparkの創設者によって開発され、大規模なデータ処理と分析のための統一された環境を提供します。この技術規範文書は、Databricksのアーキテクチャ、ベストプラクティス、設定ガイド、および使用ガイドラインを網羅し、組織がDatabricksを効果的に導入および運用するための包括的な情報を提供することを目的としています。

2. Databricks 概要

Databricksプラットフォームは、データレイクとデータウェアハウスの利点を組み合わせた「レイクハウス」アーキテクチャを基盤としています。これにより、構造化データと非構造化データの両方を単一のプラットフォームで処理・分析することが可能になります。Databricksの主要な構成要素は以下の通りです。

- **Databricksワークスペース**: ユーザーがデータにアクセスし、分析を実行し、機械学習モデルを開発するための統一された環境を提供します。ノートブック、ジョブ、モデル、実験などの管理が含まれます。
- **Apache Spark**: Databricksのコアとなる分散処理エンジンであり、大規模なデータ処理と分析を高速で実行します。DatabricksはSparkの最適化されたバージョンを提供し、パフォーマンスと信頼性を向上させています。
- **Delta Lake**: オープンソースのストレージレイヤーであり、データレイクに信頼性、セキュリティ、パフォーマンスをもたらします。ACIDトランザクション、スケーラブルなメタデータ処理、スキーマ適用、データバージョン管理などの機能を提供し、データレイクをデータウェアハウスのような信頼性で利用できるようにします。
- **MLflow**: 機械学習のライフサイクル全体（実験の追跡、再現可能な実行、モデルのデプロイ、モデルレジストリ）を管理するためのオープンソースプラットフォームです。Databricksに統合されており、データサイエンティストがモデル開発を効率的に行えるようにします。
- **Databricks SQL**: データアナリストがSQLを使用してデータレイクハウス内のデータに対して高性能なクエリを実行できるようにするサービスです。BIツールとの統合も容易です。
- **Unity Catalog**: Databricksレイクハウスのための統合されたガバナンスソリューションです。データ、AI、およびユーザーのためのきめ細かなアクセス制御、監査、およびデータリネージを提供します。

これらのコンポーネントが連携することで、Databricksはデータエンジニアリング、データサイエンス、機械学習、ビジネスインテリジェンスのワークロードを単一のプラットフォームでサポートし、データから価値を引き出すプロセスを加速します。

3. Databricks アーキテクチャ

Databricksプラットフォームは、クラウドネイティブなアーキテクチャを採用しており、コントロールプレーンとデータプレーンの2つの主要なコンポーネントで構成されています。この分離されたアーキテクチャにより、セキュリティ、スケーラビリティ、および柔軟性が向上します。

3.1. コントロールプレーン

コントロールプレーンは、Databricksのバックエンドサービスをホストし、ユーザーインターフェース、API、メタデータサービス、ジョブスケジューラ、ノートブック管理、クラスター管理などの機能を提供します。これはDatabricksによって管理され、ユーザーのクラウドアカウントとは独立して動作します。コントロールプレーンは、ユーザーのデータに直接アクセスすることはありませんが、データプレーン内のリソースをオーケストレーションし、ユーザーがDatabricksワークスペースを介してデータと対話できるようにします。

主要な機能は以下の通りです。

- **Webアプリケーション**: Databricksワークスペースへのアクセスを提供するWebベースのUI。
- **API**: プログラムによるDatabricksリソースの管理と操作を可能にするRESTful API。
- **ノートブック管理**: ノートブックの保存、バージョン管理、共有。
- **ジョブスケジューラ**: データパイプラインと機械学習ワークロードの自動化。
- **クラスター管理**: データプレーン内のコンピュートリソースのプロビジョニングと管理。
- **Unity Catalog**: データ、AI、およびユーザーのための統合されたガバナンスレイヤー。

3.2. データプレーン

データプレーンは、ユーザーのクラウドアカウント内で実行され、実際のデータ処理とストレージが行われる場所です。Databricksは、ユーザーのクラウドアカウント内にコンピュートリソース（VM、ストレージなど）をプロビジョニングし、ユーザーのデータにアクセスして処理を実行します。これにより、データはユーザーのクラウド環境内に留まり、データ主権とセキュリティが確保されます。

主要なコンポーネントは以下の通りです。

- **コンピュートリソース**: Apache Sparkクラスターを実行するための仮想マシン（VM）インスタンス。これらのクラスターは、データ処理、機械学習トレーニング、およびデータ分析ワークロードを実行します。
- **ストレージ**: Delta Lakeテーブル、生データ、およびその他のデータ資産を保存するためのクラウドストレージ（例: AWS S3, Azure Data Lake Storage, Google Cloud Storage）。
- **ネットワーク**: コントロールプレーンとデータプレーン間のセキュアな通信、およびデータプレーン内のコンピュートリソース間の通信を可能にするネットワークインフラストラクチャ。

3.3. アーキテクチャの利点

この分離されたアーキテクチャは、いくつかの重要な利点をもたらします。

- **セキュリティ**: データはユーザーのクラウドアカウント内に留まるため、データ主権とセキュリティ要件を満たしやすくなります。Databricksは、コントロールプレーンとデータプレーン間の通信をセキュアに保ちます。
- **スケーラビリティ**: コンピュートリソースは必要に応じて動的にプロビジョニングおよびスケールアップ/ダウンされるため、ワークロードの需要に応じてリソースを最適化できます。
- **柔軟性**: ユーザーは、独自のクラウドアカウントと既存のクラウドインフラストラクチャを活用できます。
- **分離**: コントロールプレーンとデータプレーンの分離により、Databricksの管理サービスとユーザーのデータ処理環境が明確に分離され、運用上の複雑さが軽減されます。

以下にDatabricksのアーキテクチャの概念図を示します。

4. Databricks ベストプラクティス

Databricksを最大限に活用し、効率的かつセキュアなデータおよびAIプラットフォームを構築するためには、以下のベストプラクティスを遵守することが重要です。

4.1. データレイクハウスの設計

- **メダリオンアーキテクチャの採用**: データをブロンズ（生データ）、シルバー（クリーンでフィルタリングされたデータ）、ゴールド（集計され、ビジネスロジックが適用されたデータ）の3層に分割するメダリオンアーキテクチャを採用することで、データの品質と信頼性を段階的に向上させ、データガバナンスを強化します。
- **Delta Lakeの活用**: ACIDトランザクション、スキーマ適用、タイムトラベルなどのDelta Lakeの機能を活用し、データレイクの信頼性とパフォーマンスを向上させます。
- **Unity Catalogによるデータガバナンス**: Unity Catalogを使用して、データ、テーブル、モデルに対するきめ細かなアクセス制御、監査、およびデータリネージを実装し、データセキュリティとコンプライアンスを確保します。

4.2. パフォーマンス最適化

- **クラスターの適切なサイジング**: ワークロードの要件に基づいて、クラスターのタイプ、インスタンス数、およびインスタンスタイプを適切に選択します。コストとパフォーマンスのバランスを考慮します。
- **データのパーティショニングとZ-Ordering**: 大規模なテーブルに対しては、クエリのパフォーマンスを向上させるために、データのパーティショニングとZ-Ordering（Delta Lakeの最適化機能）を適用します。
- **キャッシュの活用**: 頻繁にアクセスされるデータに対しては、Databricksのディスクキャッシュ やSparkのキャッシュ機能を活用して、クエリの実行速度を向上させます。
- **最適化されたファイル形式**: ParquetやDelta Lakeなどの列指向ストレージ形式を使用し、データの読み込みと処理の効率を高めます。

4.3. 開発と運用（DevOps）

- **バージョン管理の統合**: ノートブックやコードをGitなどのバージョン管理システムと統合し、共同開発、変更履歴の追跡、および再現性を確保します。
- **CI/CDパイプラインの構築**: Databricks ReposやDatabricks Workflowsを活用して、データパイプラインや機械学習ワークロードの継続的インテグレーションと継続的デリバリー（CI/CD）パイプラインを自動化します。
- **テストの自動化**: データ品質テスト、ユニットテスト、統合テストを自動化し、データパイプラインと機械学習モデルの信頼性を確保します。

- **モニタリングとアラート**: Databricksのログ、メトリクス、および外部のモニタリングツール（例: Prometheus, Grafana）を使用して、ワークロードのパフォーマンスと健全性を監視し、問題が発生した際にはアラートを生成します。

4.4. セキュリティとコンプライアンス

- **最小権限の原則**: ユーザーとサービスプリンシパルには、その役割を果たすために必要な最小限の権限のみを付与します。
- **ネットワークセキュリティ**: VPC/VNetピアリング、プライベートリンク、ファイアウォールルールなどを設定し、Databricks環境へのネットワークアクセスを制限します。
- **データ暗号化**: 保存データ（at rest）と転送データ（in transit）の両方でデータ暗号化を有効にし、データの機密性を保護します。
- **監査ログの有効化**: Databricksの監査ログを有効にし、すべての操作を記録することで、セキュリティインシデントの調査とコンプライアンス要件の遵守を支援します。

これらのベストプラクティスを適用することで、Databricks環境のパフォーマンス、信頼性、セキュリティ、および管理性を向上させることができます。

5. Databricks 設定ガイド

Databricks環境の適切な設定は、パフォーマンス、セキュリティ、および管理の効率性に直結します。以下に、主要な設定項目と推奨されるガイドラインを示します。

5.1. ワークスペースのセットアップ

- **クラウドプロバイダーの選択**: AWS、Azure、Google Cloudの中から、組織の既存のクラウドインフラストラクチャと戦略に合致するプロバイダーを選択します。
- **リージョンの選択**: ユーザーの地理的位置、データ所在地要件、および利用可能なサービスに基づいて、適切なクラウドリージョンを選択します。
- **VPC/VNetの構成**: Databricksワークスペースをデプロイする前に、既存のVPC（AWS/GCP）またはVNet（Azure）を適切に構成し、必要なネットワーク接続とセキュリティグループを設定します。プライベートサブネットの使用を推奨します。

- **ストレージアカウントの準備**: Delta Lakeテーブルやその他のデータを保存するためのクラウドストレージアカウント（S3バケット、ADLS Gen2、GCSバケットなど）を準備し、Databricksワークスペースからのアクセス権限を設定します。

5.2. クラスターの設定

- **クラスターモード**: ワークロードの性質に応じて、以下のクラスターモードを選択します。
    - **標準クラスター**: 複数のユーザーがインタラクティブに作業する場合に適しています。
    - **ジョブクラスター**: 自動化されたジョブやETLパイプラインの実行に最適化されています。ジョブの実行が完了すると自動的に終了するため、コスト効率が高いです。
    - **高コンカレンシークラスター**: 複数のユーザーが同時に同じクラスターで作業する場合に、リソースの分離とパフォーマンスの安定性を提供します。
- **インスタンスタイプとサイズ**: ワークロードの計算要件（CPU、メモリ、GPUなど）とコストを考慮して、適切なインスタンスタイプとサイズを選択します。データ処理量や複雑性に応じて、ワーカーノードの数を調整します。
- **Databricks Runtimeバージョン**: ワークロードに必要なライブラリ、Sparkバージョン、および最適化が含まれるDatabricks Runtimeバージョンを選択します。最新のLTS（長期サポート）バージョンを使用することを推奨します。
- **自動スケーリング**: クラスターの自動スケーリングを有効にすることで、ワークロードの需要に応じてワーカーノードの数を自動的に調整し、リソースの最適化とコスト削減を実現します。
- **自動終了**: アイドル状態のクラスターが自動的に終了するように設定し、不要なコストの発生を防ぎます。
- **Initスクリプト**: クラスターの起動時にカスタムライブラリのインストールや設定の適用が必要な場合は、Initスクリプトを使用します。

5.3. セキュリティ設定

- **認証と認可**:
    - **SSO（シングルサインオン）の統合**: 組織の既存のIDプロバイダー（Azure AD, Oktaなど）とSSOを統合し、ユーザー認証を一元化します。

    - **Unity Catalogによるアクセス制御**: Unity Catalogを使用して、データ、テーブル、ビュー、関数、および機械学習モデルに対するきめ細かなアクセス制御を設定します。最小権限の原則に従い、必要な権限のみを付与します。
- **ネットワーク分離**:
    - **VPC/VNetピアリング**: オンプレミスネットワークや他のクラウドVPC/VNetとのセキュアな接続を確立します。
    - **プライベートリンク**: Databricksコントロールプレーンとデータプレーン間の通信、およびDatabricksとクラウドストレージ間の通信をプライベートネットワーク経由で行うように設定し、データの漏洩リスクを低減します。
- **データ暗号化**:
    - **保存データの暗号化**: クラウドストレージに保存されるデータは、クラウドプロバイダーの暗号化サービス（SSE-S3, SSE-KMSなど）または顧客管理キー（CMK）を使用して暗号化されていることを確認します。
    - **転送データの暗号化**: Databricksクラスター内のノード間通信や、Databricksと外部サービス間の通信は、TLS/SSLを使用して暗号化されていることを確認します。
- **監査ログ**: Databricksの監査ログを有効にし、すべてのユーザーアクティビティとシステムイベントを記録します。これにより、セキュリティインシデントの調査、コンプライアンス要件の遵守、および不正アクセスの検出が可能になります。

これらの設定ガイドラインに従うことで、Databricks環境をセキュアかつ効率的に運用するための強固な基盤を構築できます。

6. Databricks 使用ガイドライン

Databricksを効果的に使用するためには、開発者、データサイエンティスト、データアナリストが以下のガイドラインを遵守することが重要です。これにより、コードの品質、パフォーマンス、コラボレーション、および保守性が向上します。

6.1. ノートブックの使用

- **モジュール化と再利用性**: ノートブックを論理的なセクションに分割し、再利用可能な関数やクラスを定義して、コードの重複を避け、保守性を高めます。共通の処理は、Databricks Reposに保存されたPythonやScalaのモジュールとして管理することを検討します。

- **コメントとドキュメンテーション**: コードの目的、ロジック、および重要な仮定を明確にするために、適切なコメントとMarkdownセルを使用してノートブックをドキュメント化します。特に、複雑な処理やビジネスロジックについては詳細な説明を含めます。
- **バージョン管理**: ノートブックはDatabricks Reposを介してGitと統合し、変更履歴を追跡し、共同開発を容易にします。コミットメッセージは、変更内容を明確に記述するようにします。
- **エラーハンドリングとロギング**: ノートブック内でエラーハンドリング（try-exceptブロックなど）を実装し、予期せぬエラーが発生した場合に適切に処理します。重要な処理の実行状況やエラー情報をログに出力し、デバッグと監視を容易にします。
- **出力の管理**: ノートブックの出力は、過度に大きくならないように注意します。大規模なデータフレームの表示や、不必要なグラフの生成は、ノートブックのロード時間を増加させ、パフォーマンスに影響を与える可能性があります。必要に応じて、`display()`関数や`limit()`関数を使用して出力を制限します。

6.2. データ処理と分析

- **Delta Lakeの活用**: データの取り込み、変換、および分析には、Delta Lakeを使用することを推奨します。Delta LakeのACIDトランザクション、スキーマ適用、およびタイムトラベル機能は、データパイプラインの信頼性と堅牢性を高めます。
- **ストリーミング処理**: リアルタイムまたはニアリアルタイムのデータ処理が必要な場合は、Apache Spark Structured StreamingとDelta Lakeを組み合わせて使用します。これにより、増分データ処理と低レイテンシの分析が可能になります。
- **データ品質の確保**: データパイプラインの各段階でデータ品質チェックを実装し、データの整合性と正確性を確保します。期待されるスキーマからの逸脱、欠損値、異常値などを検出するメカニズムを導入します。
- **SQLとPython/Scalaの適切な使い分け**: データ変換や分析には、SQLとPython/Scala（PySpark/Spark Scala）を適切に使い分けます。SQLはデータ探索やシンプルな変換に適しており、Python/Scalaは複雑なロジック、機械学習、およびカスタム関数の開発に適しています。
- **コスト最適化**:
    - **クラスターの自動終了**: 不要なクラスターが実行され続けないように、自動終了時間を適切に設定します。

    - **スポットインスタンスの利用**: 開発環境やフォールトトレラントなワークロードでは、コスト削減のためにスポットインスタンスの利用を検討します。
    - **効率的なコード**: データのシャッフルを最小限に抑え、ブロードキャストジョインを活用するなど、Sparkのパフォーマンス最適化のベストプラクティスに従ってコードを記述します。

6.3. 機械学習ワークフロー

- **MLflowの活用**: 機械学習モデルの開発、追跡、デプロイにはMLflowを使用します。実験のパラメータ、メトリクス、およびモデルアーティファクトを追跡し、再現可能な機械学習ワークフローを構築します。
- **特徴量エンジニアリング**: 特徴量ストア（Databricks Feature Storeなど）を活用して、特徴量の再利用性を高め、特徴量エンジニアリングのプロセスを効率化します。
- **モデルのバージョン管理とレジストリ**: MLflow Model Registryを使用して、モデルのバージョン管理、ステージング（開発、ステージング、本番）、および承認ワークフローを管理します。
- **モデルのデプロイ**: バッチ推論、ストリーミング推論、またはリアルタイム推論の要件に応じて、適切なモデルデプロイ戦略（Databricks Model Serving, MLflowによるカスタムデプロイなど）を選択します。

これらの使用ガイドラインに従うことで、Databricksプラットフォームを最大限に活用し、データとAIのプロジェクトを成功に導くことができます。

7. まとめ

この文書では、Databricksプラットフォームの包括的な技術規範を提供しました。Databricksは、データエンジニアリング、データサイエンス、機械学習、およびビジネスインテリジェンスのワークロードを統合する強力なレイクハウスプラットフォームです。そのユニークなアーキテクチャ、豊富な機能セット、およびクラウドネイティブな設計により、組織はデータから迅速に価値を引き出し、AI主導のイノベーションを加速することができます。

本規範で概説したアーキテクチャの原則、ベストプラクティス、設定ガイド、および使用ガイドラインを遵守することで、組織はDatabricks環境を最適化し、セキュリティを強化し、運用効率を向上させることができます。これにより、データチームはより信頼性の高

いデータパイプラインを構築し、より正確な機械学習モデルを開発し、より深い洞察を導き出すことが可能になります。

Databricksの導入と運用は継続的なプロセスであり、技術の進化と組織のニーズの変化に合わせて、本規範も定期的に見直し、更新することが重要です。この文書が、Databricksを最大限に活用し、データ駆動型組織への変革を成功させるための一助となることを願っています。