PROJETO DATABRICKS SEGURO

RELATÓRIO FINAL DO PROJETO

Sumário

1 Resumo

Esse projeto é uma implementação de um padrão arquitetural de acesso seguro aos dados armazenados em um Azure Data Lake Storage (ADSL) por notebooks desenvolvidos no Azure Databricks, utilizando um *service principal* como um mediador do acesso. Sua principal função é demonstrar uma forma de acesso a esses dados por meio dos notebooks, utilizando uma sintaxe de pontos de montagem e sem expor informações sensíveis do ambiente Azure nos notebooks acessados pelos desenvolvedores. A solução destina-se a engenheiros de dados e cientistas de dados que utilizam a plataforma Azure Databricks para seus projetos de ciência de dados e que precisam acessar dados de um ADLS. Devido ao rápido desenvolvimento do Azure Databricks, estima-se que outras soluções, como o Unity Catalog, venham a substituir a arquitetura apresentada nesse projeto nos próximos anos, como uma alternativa de controle de acesso seguro aos dados.

2 Descrição e objetivos gerais

Esse projeto visa facilitar a implantação de um padrão arquitetural de acesso seguro aos dados armazenados em um Azure Data Lake Storage por notebooks desenvolvidos na plataforma Azure Databricks.

Com o uso de um mediador (*service principal*) é possível estabelecer uma conexão segura do Azure Databricks com vários recursos do Azure, incluindo o Data Lake Storage, um recurso de armazenamento de grandes volumes de dados, muito utilizado como fonte de dados em projetos de *Data Science*.

Integrar o Azure Databricks ao Azure Data Lake Storage (ADLS) traz vários benefícios devido à natureza complementar desses serviços. Podemos destacar os seguintes ganhos ao estabelecer essa conexão:

Escalabilidade e Performance: O ADLS é projetado para escalabilidade e alta velocidade, o que significa que pode lidar com grandes conjuntos de dados e cargas de trabalho de big data. O Azure Databricks, sendo uma plataforma de análise baseada em Apache Spark, pode aproveitar essa escalabilidade e performance para processar dados em grande escala rapidamente.

Segurança Avançada: A integração permite aproveitar os recursos de segurança do ADLS, como controle de acesso baseado em função (RBAC), autenticação via Azure Active Directory e criptografia de dados em repouso e em trânsito. Com o Azure Databricks, você pode estabelecer políticas de acesso granulares para garantir que apenas usuários autorizados possam acessar e modificar dados.

Gerenciamento Unificado de Dados: Ao integrar o Azure Databricks ao ADLS, você centraliza seus dados em um único repositório, tornando mais fácil gerenciar, monitorar e governar seus dados em um local unificado.

Economia de Custos: O Azure Databricks e o ADLS são otimizados para a computação em nuvem, permitindo escalabilidade sob demanda. Isso significa que você paga apenas pelo que usa, sem necessidade de provisionar infraestrutura cara de antemão.

Integração e Conectividade: A combinação permite fácil integração com outros serviços do Azure e ferramentas de terceiros. Isso facilita a construção de pipelines de dados end-to-end, desde a ingestão até a análise e visualização.

Formatos de Dados Flexíveis: O ADLS suporta vários formatos de dados, de arquivos de texto simples a parquet e avro. O Azure Databricks, com suas bibliotecas incorporadas, pode processar esses diferentes formatos facilmente, oferecendo flexibilidade na análise de dados.

Confiabilidade e Disponibilidade: O ADLS oferece armazenamento durável com capacidades de recuperação de desastres. Em combinação com a confiabilidade do Azure Databricks, isso garante que seus pipelines de dados e análises estejam sempre disponíveis e resistentes a falhas.

Desenvolvimento Colaborativo: O Azure Databricks fornece notebooks colaborativos que permitem que analistas de dados, engenheiros de dados e cientistas de dados colaborem em projetos. Com os dados centralizados no ADLS, as equipes podem trabalhar juntas de forma mais eficiente.

3 Requisitos funcionais e não funcionais

Requisitos funcionais:

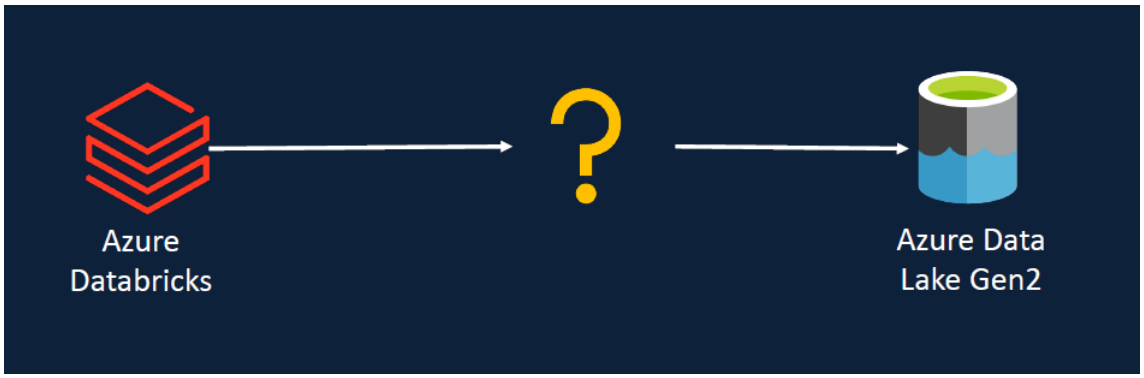
- A solução deve garantir um acesso seguro aos dados do Data Lake Storage pelos notebooks do Azure Databricks;
- A solução deve dispor de uma classe em Python para acesso facilitado ao Data Lake Storage;
- O acesso aos dados deve ser feito por meio de pontos de montagem, usando uma semântica de acesso a arquivos e pastas.

Requisitos não funcionais:

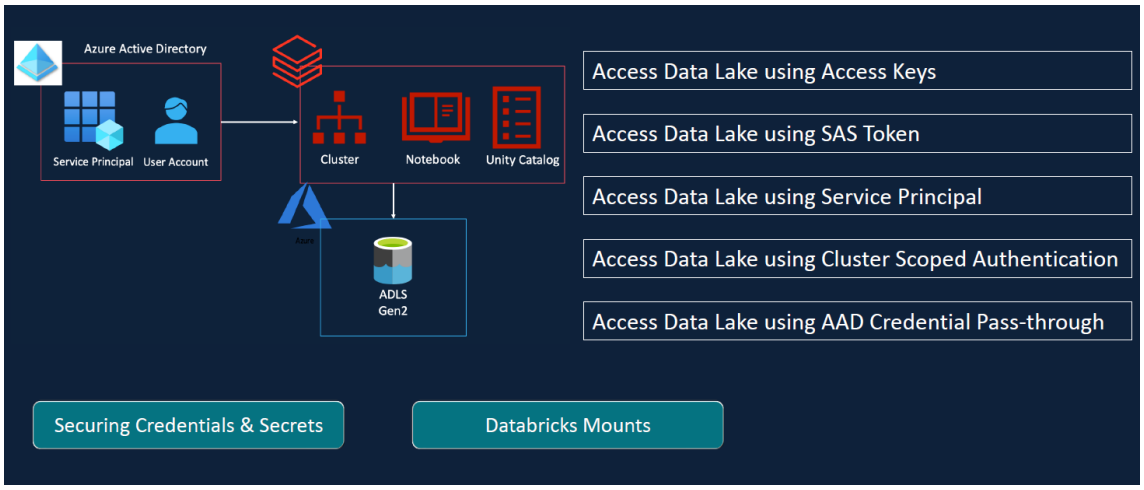
- Uma conta de armazenamento para criação do ADSL (Azure Data Lake Storage);
- Um cofre de chaves (Azure Key Vault);
- Um workspace do Azure Databricks;
- Um cluster do Azure Databricks.

4 Modelos de arquitetura

A principal motivação desse projeto é responder a pergunta: Como prover um acesso seguro dos notebooks do Azure Databricks aos dados em um Azure Data Lake Storage?



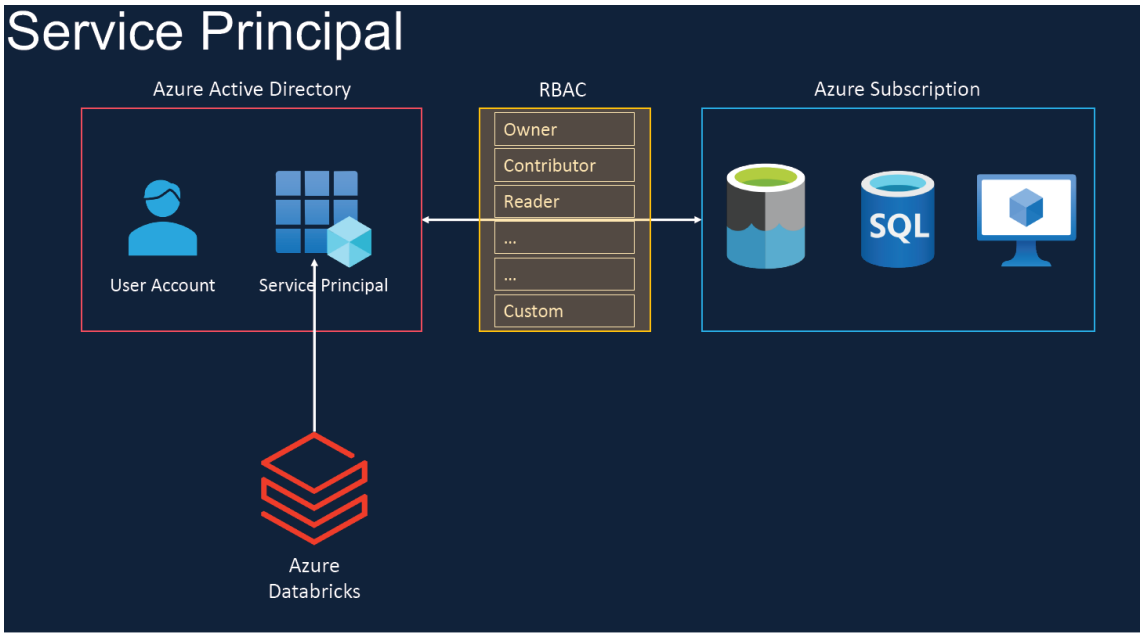
Existem diversos padrões arquiteturais para viabilizar esse acesso, conforme ilustra a figura a seguir.



De todos os padrões disponíveis, aquele que garante maior segurança no acesso é o uso do Service Principal, conforme será detalhado no item a seguir.

4.1 Service Principal

Um Service Principal é um recurso muito semelhante a uma conta de usuário no Azure. Ela é registrada no Azure Active Directory e tem associada a ela permissões de acesso aos recursos de uma assinatura do Azure por meio do RBAC (Role Based Access Control). É o método recomendado para ser usado em *jobs* e *pipelines* de processamento do Azure Databricks, pois fornece uma maior segurança e rastreabilidade. Em uma boa arquitetura, cada aplicação deve ter o seu próprio Service Principal, e cada Service Principal apenas as permissões necessárias para a aplicação ao qual está vinculado precise ter.



Para implementar um Service Principal na arquitetura proposta, são necessárias as seguintes etapas:

- Registrar o Service principal (ou Aplicação) no Azure Active Directory
- Gerar um segredo para a Aplicação
- Configurar o Spark com ID do Cliente, ID do Tenant e o segredo
- Associar o papel “Storage Blob Data Contributor ou Reader” para o Data Lake, para que o Service Principal acesse os blobs do data lake.

4.2 DBFS e Databricks Mounts

O Databricks File System (DBFS) é uma camada de abstração que simplifica o acesso ao armazenamento de objetos, proporcionando uma experiência semelhante a um sistema de arquivos para os usuários e aplicações do Databricks. Ele desempenha um papel fundamental na integração do Azure Databricks com as estruturas de armazenamento disponíveis no Azure, como o Data Lake Storage. Como principais características, podemos destacar:

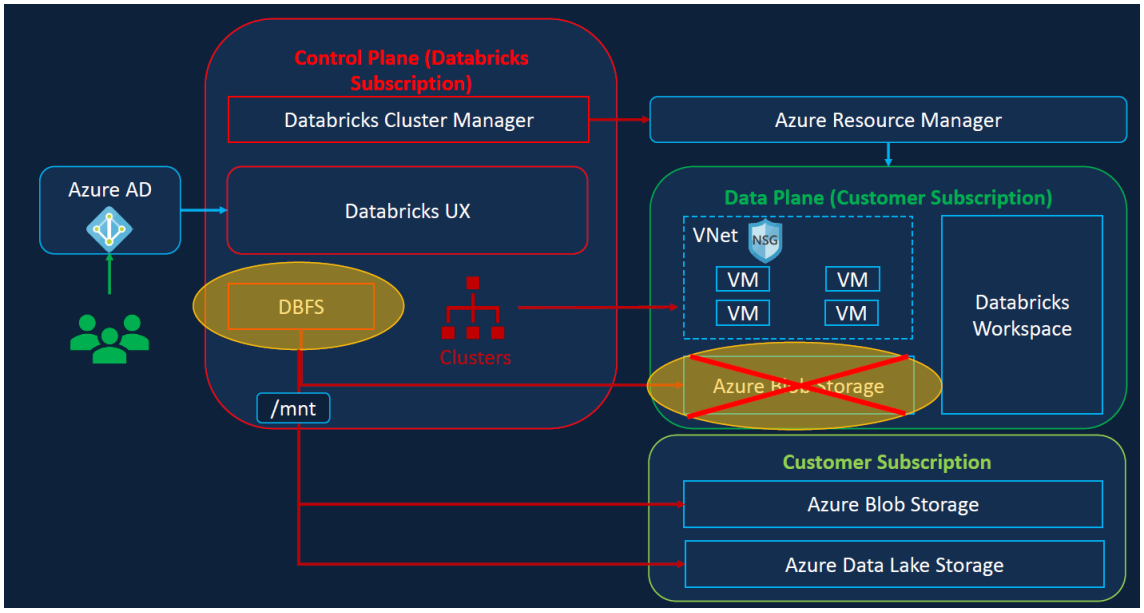
- Através do DBFS, os usuários e aplicações podem interagir com os dados armazenados como se estivessem trabalhando com um sistema de arquivos local ou HDFS. Isso simplifica a leitura e escrita de dados no armazenamento de objetos.
- o DBFS armazena dados no Azure Blob Storage, que oferece durabilidade e economia de custos. O DBFS torna essa interação transparente para o usuário.
- Você pode "montar" containers de armazenamento ou diretórios do Azure Blob Storage em pontos específicos do DBFS. Isso permite que você acesse seus dados no Azure Blob Storage através de caminhos no DBFS.
- Como o DBFS é construído sobre o Azure Blob Storage, ele herda suas características de durabilidade e disponibilidade. Além disso, o DBFS garante consistência de leitura após gravação, o que é essencial para cargas de trabalho de análise de dados.
- O DBFS é otimizado para suportar streaming e grandes cargas de trabalho de análise de dados. Ele pode lidar com grandes volumes de dados e suporta operações de leitura e gravação em paralelo.

- O DBFS funciona bem com as ferramentas e frameworks do ecossistema Hadoop/Spark. Os dados no DBFS podem ser lidos diretamente por aplicativos Apache Spark, facilitando o processamento e a análise.

Ainda que possa armazenar dados, não se recomenda o uso do DBFS para armazenamento de dados da organização, pois caso o espaço de trabalho seja desativado, os dados são perdidos. Nesse sentido, Databricks permite a montagem de uma conta de armazenamento como um ponto de montagem no DBFS (Databrick mount). As credenciais de acesso são fornecidas no momento da montagem. Feito isso, todos que tiverem acesso a workspace terão acesso aos dados da conta de armazenamento que foi montada. O acesso aos dados passa a ser feito por uma semântica de arquivos e pastas, diferente das longas URLs.

Os principais benefícios dessa abordagem são:

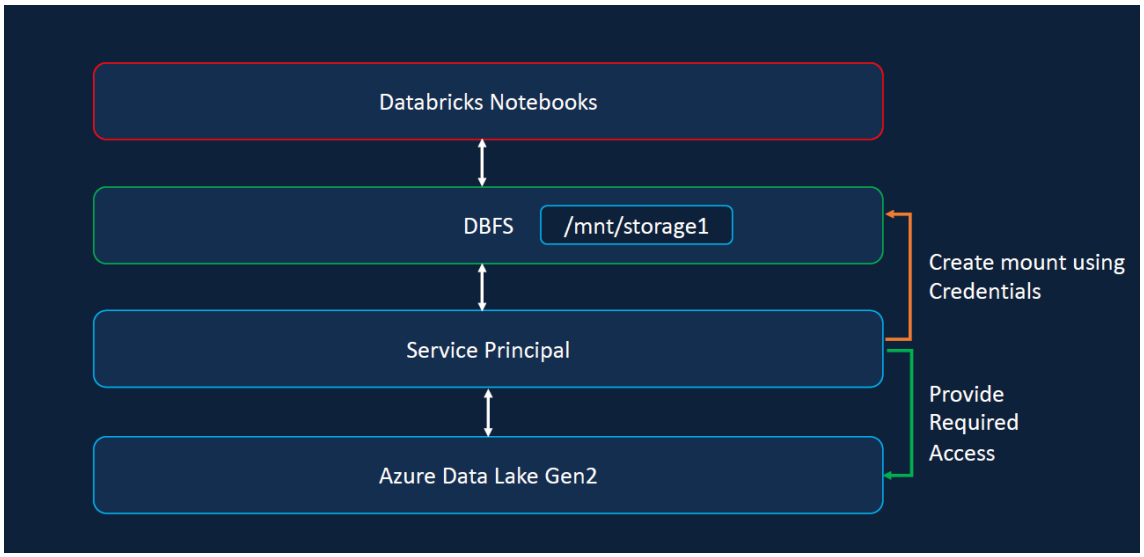
- Acesso aos dados sem precisar das credenciais;
- Acesso aos arquivos usando uma semântica de pastas ao invés de URLs (e.g. /mnt/storage1)
- Armazena arquivos em objetos do Blob Storage.



Vale destacar que o recurso "Unity Catalog" vem sendo adotado como padrão para gerenciamento de acesso aos dados nos projetos mais recentes. O Unity Catalog é a solução de governança de dados do Databricks, oferecendo a capacidade de gerenciar a disponibilidade, usabilidade, integridade e segurança dos dados presentes em uma organização. Como esse produto foi lançado no final de 2022, sua adoção ainda não é plena e os padrões arquiteturais apresentados anteriormente ainda são os mais utilizados em projetos que envolvam o uso do ADSL e do Azure Databricks.

4.3 Solução final de acesso

Com o uso do Service Principal e do DBFS, é possível implementar o acesso aos dados de forma simples e segura, conforme a figura a seguir.



Os dados armazenados em contêineres de blobs no Azure Data Lake Gen2 são disponibilizados em pontos de montagem no DBFS. A mediação do controle de acesso é garantida pelo Service Principal. Os notebooks passam a ter acesso aos dados por meio dos pontos de montagem no DBFS, sem precisar expor os dados sensíveis do locatário em texto claro nos notebooks.

4.4 Arquitetura Medallion

Outra arquitetura adotada pelo projeto é a arquitetura *Medallion*. A Arquitetura *Medallion* é uma abordagem proposta pelo Databricks para construir modernos pipelines de análise de dados em escala utilizando o ecossistema do Databricks e do Apache Spark. Essa abordagem é centrada na ideia de criar camadas de dados, ou "medalhões", para otimizar a eficiência, o desempenho e a simplicidade dos pipelines de dados.

A Arquitetura *Medallion* consiste geralmente em três camadas:

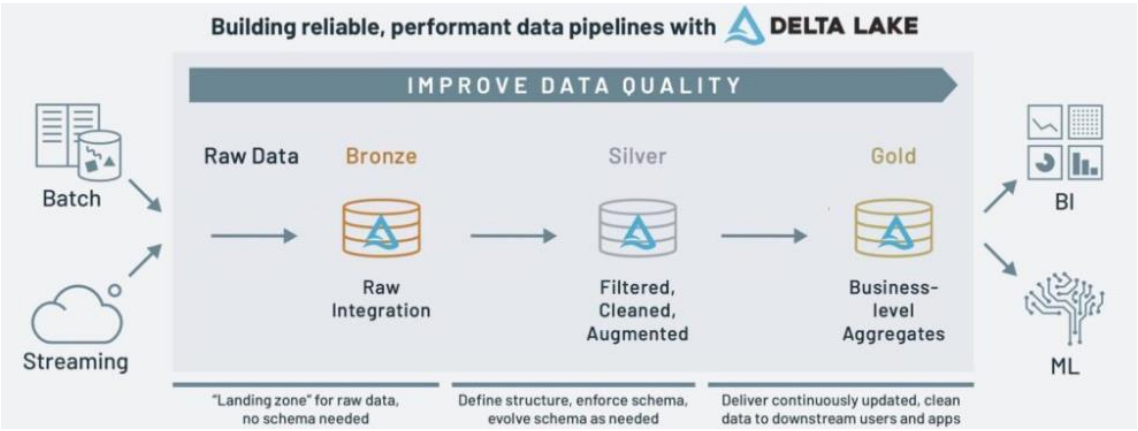
- Camada Bronze (*Raw Data Layer*): Esta camada armazena os dados em seu formato bruto, muitas vezes como são ingeridos a partir de fontes de dados originais. Serve como uma zona de aterrissagem para os dados e frequentemente armazena dados em formatos semiestruturados, como JSON ou XML. Mantendo os dados em seu formato original, as organizações têm um registro histórico completo, o que é útil para auditorias, recuperação de desastres ou reprocessamento.
- Camada Prata (*Curated Data Layer*): Nesta camada, os dados são processados, limpos, enriquecidos e transformados em um formato mais utilizável e otimizado. Pode envolver operações como limpeza de dados, correção de erros, deduplicação e aplicação de regras de negócios. Como os dados nesta camada são limpos e otimizados, é mais eficiente para o consumo por cientistas de dados e analistas. Também serve como uma base confiável para a criação da camada Ouro.
- Camada Ouro (*Consumable Data Layer*): Esta camada armazena os conjuntos de dados que são preparados especificamente para o consumo final. Isso pode envolver a agregação de dados, a criação de recursos ou a transformação de dados em formatos específicos para relatórios ou modelagem. Isso facilita o consumo rápido e eficiente de

dados para usuários finais, seja para análise de BI, modelagem de machine learning ou outras tarefas analíticas.

Vale destacar que o Delta Lake, uma solução de armazenamento em camadas construída sobre o Apache Spark e desenvolvida pelo Databricks, é frequentemente usado em conjunto com a Arquitetura *Medallion* para otimizar o armazenamento, a qualidade e o desempenho dos dados.

Por fim, a abordagem *Medallion* ajuda as organizações a gerenciar dados em escala, garantindo que os dados estejam prontamente disponíveis em diferentes estágios do pipeline de análise, desde a ingestão bruta até o consumo final.

O Data Lake provisionado pela solução segue essa arquitetura, criando containers que mapeiam as camadas apresentadas.



5 Descrição do software

A solução desenvolvida no âmbito desse projeto permite a rápida implementação de um ambiente de experimentação e desenvolvimento de soluções usando o Azure Databricks acessando dados de um Azure Data Lake Storage Gen2, de forma simples e segura.

A solução permite um provisionamento automático, via infraestrutura como código, dos recursos necessários para utilização desse ambiente em um dado tenant do Azure.

A solução dispõe ainda de código em Python para validar a implantação, por meio da criação dos pontos de montagem para acesso aos containers disponíveis, pelos notebooks.

6 Manual de utilização

Deve ser utilizado o manual de instalação do ambiente, que tem o passo a passo para implantação do ambiente e o código para validação da instalação.

7 Cenários de uso

7.1 Cenário 1: Estudante de computação

Um estudante de computação pode criar um tenant com créditos gratuitos do Azure e provisionar um ambiente do Azure Databricks nesse tenant, para experimentações dessa tecnologia.

A execução dos passos de instalação é simples e direta. É desejável que o estudante tenha um conhecimento mínimo do Azure e do Portal.

7.2 Cenário 2: Engenheiro de dados em uma Organização

Um engenheiro de dados precisa ter acesso a um ambiente do Azure Databricks em sua empresa. Ele pode encaminhar essa solução ao time de infraestrutura da Organização, que pode utilizá-lo para provisionar o ambiente de forma expedita.

O script foi pensado para ser o menos intrusivo possível no ambiente da Organização. Os recursos criados de forma automatizada são os mínimos necessários para o funcionamento da solução. Todos os detalhes que requeiram o uso de padrões específicos adotados pelo time de infraestrutura da organização são parametrizáveis ou podem ser ajustados após o provisionamento, seguindo a documentação em anexo desse projeto.

8 Cenários adversos

8.1 Cenário 1: Falta de experiência com o ambiente Azure

Ainda que boa parte do processo de implantação seja automatizado, parte dele requer alguma capacidade de interação com o Portal do Azure. Como alguns passos da implantação requerem configurações por meio desse Portal, uma operação errada pode comprometer toda a implantação.

8.2 Cenário 2: Bibliotecas do Powershell desatualizadas

Parte do código de implantação está escrito em Powershell. A Microsoft frequentemente atualiza as bibliotecas, o que pode implicar em mudanças no código e erros na implantação.