Lab 1 - Use Tools to optimize Power BI Performance

Note: Alternatively, you can follow this other lab: <https://learn.microsoft.com/en-us/training/modules/optimize-model-power-bi/>). Some steps are the same, some are complementary. Portuguese version: [Otimizar um modelo para desempenho no Power BI - Training | Microsoft Learn](https://learn.microsoft.com/pt-br/training/modules/optimize-model-power-bi/)

# Introduction

You've built Power BI reports and they're running slow. How do you identify the source of the problem? The report could be slow because of issues at the *source*, the structure of the *data model*, the *visuals* on the report page, or the *environment*.

Optimizing performance in the context of this module means making changes to the *data model* so that it runs more efficiently. The tools we're discussing in this module will help you troubleshoot and improve the data model, which ultimately improves user experience.

Ideally, you should assess performance as you move through the stages of the development of lifecycle, development, testing, production, and optimization of your solutions.

In this module you'll learn how to use tools to optimize a data model, and which tools are useful in which scenarios.

**Learning objectives**

After completing this module, you'll be able to:

* Optimize queries using Performance analyzer.
* Troubleshoot DAX performance using DAX Studio.
* Optimize a data model using Tabular Editor.

# Pre-requirement

Power BI Desktop installed in your notebook.

Take a report that you have already built before to use your analysis.

# Use Performance analyzer

The first tool we're looking at is the [Performance analyzer](https://learn.microsoft.com/en-us/power-bi/create-reports/desktop-performance-analyzer), which is built in to Power BI Desktop. The Performance analyzer helps you understand how report elements like visuals and DAX queries are performing.

The Performance analyzer helps you optimize at two of the four architecture levels, the *data model* and *report visuals*. The Performance analyzer is a great place to start when you're optimizing reports.

**Understand performance using Performance analyzer**

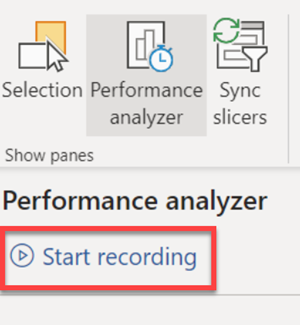
The Performance analyzer displays and records logs that measure how each of your report elements performs when users interact with them. You can also see which aspects of their performance are most (or least) resource intensive. You can immediately see results, make changes, run the analyzer again, and see the impact of changes you've made.

The Performance Analyzer captures operations that occur in several major subsystems involved in executing a Power BI Report:

* **Report Canvas** provides the user interface for Power BI reports including hosting visuals and filters, managing user interactions for consuming and authoring reports, and retrieving data for display. The Report Canvas is written using web technologies and runs in web browsers or web browser components. The Report Canvas retrieves data using a high-level, internal, Power BI query language known as *Semantic Query*.
* **Data Shape Engine (DSE)** evaluates Semantic Queries by generating and running one, or more DAX queries against a data model hosted inside Power BI, Power BI Desktop, Azure Analysis Services, or SQL Server Analysis Services.
* **Data Model Engine (AS)** stores the data model and provides services to reports, such as DAX query evaluation. The model may be hosted in Power BI, Power BI Desktop, Azure Analysis Services, or SQL Server Analysis Services. Depending on the data model host, a model may be tabular or multidimensional. Tabular models may contain in-memory tables, Direct Query tables, or a mix of such tables. DAX queries against tables in Direct Query mode will trigger queries to the Direct Query data source. For example, a DAX query against a Direct Query table backed by a SQL Server database will trigger one, or more, SQL queries.

**Use Performance analyzer**

To use the Performance analyzer, enable the Performance analyzer on the View tab of the ribbon, and select **Start recording**.

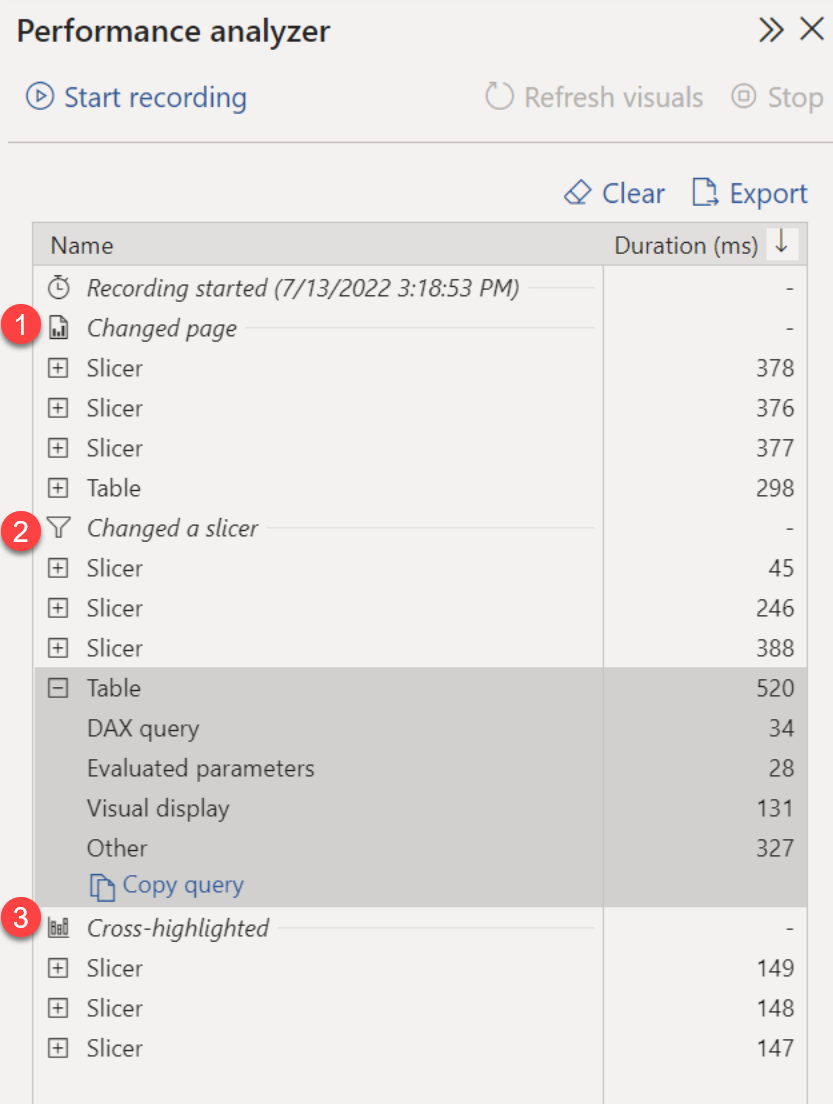


After you start recording, take actions in the report. For example, move from one report tab to the next, select a slicer item, or interact with any of your visuals. Any actions you take in the report are displayed and logged in real time in the Performance Analyzer pane, in the order that the visual is loaded by Power BI.

The Performance analyzer is looking at the time it takes for each visual to query the data model and render results. This is the time from when a user does something on the page to when the visual is rendered, in milliseconds.

In the example below, three actions were taken after recording started.

1. Navigated to a new report page
2. Changed a slicer value
3. Cross highlighted a table



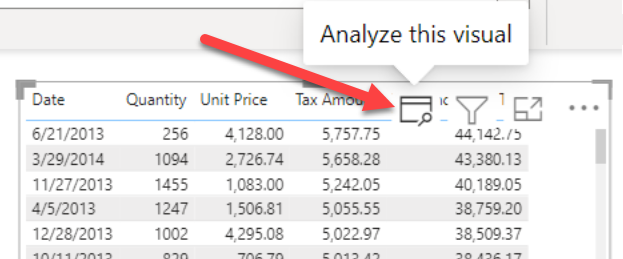
Each visual's log information includes the time spent (duration) completing the following categories of tasks:

* **DAX query** - if a DAX query was required, this is the time between the visual sending the query, and for Analysis Services to return the results.
* **Evaluated parameters (preview)** - time spent evaluating the field parameters within a visual. Learn more about [field parameters (preview)](https://learn.microsoft.com/en-us/power-bi/create-reports/power-bi-field-parameters" \t "_blank).
* **Visual display** - time required for the visual to appear on the screen, including time required to retrieve any web images or geocoding.
* **Other**- time required by the visual for preparing queries, waiting for other visuals to be complete, or performing other background processing.

After you've interacted with elements of the report you want to measure with Performance Analyzer, you can select the Stop button. The performance information remains in the pane after you select Stop for you to analyze.

You can select **Refresh visuals** in the Performance Analyzer pane to refresh all visuals on the current page of the report, which will gather information about all visuals.

If one particular visual appears to be performing slow, you can also refresh individual visuals. When Performance Analyzer is recording, you can select **Analyze this visual** found in the top-right corner of each visual, to refresh that visual, and capture its performance information.



**Important**

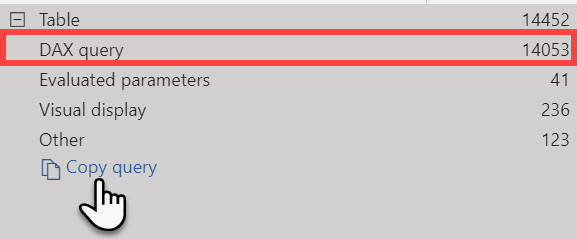
As users interact with visuals in Power BI reports, DAX queries are submitted to the dataset and cached into memory. Because of this, you may need to clear the DAX query cache to get accurate results in the Performance analyzer. You can clear the cache by either closing and re-opening your report or using DAX Studio.

**Evaluate performance data further**

There are many ways you can dig deeper into information recorded by the Performance analyzer.

You can use DAX Studio to investigate your queries in more detail by copying your query from the performance analyzer. After analyzing the query in DAX Studio, you can use your own knowledge and experience to identify where the performance issues are.

In the screenshot below, you can see that the DAX query in this table visual took 14 seconds. Users waited 14 seconds before seeing the results of the action that resulted in that query running, which in this case was a cross-highlight action. It's clear that this DAX needs to be optimized.



To analyze this query further, copy and paste the query into DAX Studio to repeat the execution of the query. In DAX Studio, you can activate more diagnostic tools using the Query Plan and Server Timings tracing options.

For DAX queries with long duration times, it's likely that a measure is written poorly or an issue has occurred with the data model. The issue might be caused by the relationships, columns, or metadata in your model, or it could be the status of the Auto date/time option.

You can also save Performance analyzer results by selecting the **Export**button, which creates a .json file containing results. Each event in the .json file contains timestamps, correlation information, and other metadata about the operation.

You can then load that data into DAX Studio to navigate through the performance metrics in more detail.

**Tip**

For more information about the contents of the .json export file, see the [Performance analyzer export file documentation](https://github.com/microsoft/powerbi-desktop-samples/commits/main/Performance%20Analyzer/Power%20BI%20Performance%20Analyzer%20Export%20File%20Format.docx).

# Troubleshoot DAX performance by using DAX Studio

In unit 2, we learned that Performance analyzer data can be further analyzed in DAX Studio. Import performance data into DAX Studio, where you can view, sort, and filter all performance data. You can also bring a single query into the editor where you can make adjustments to the query to improve performance.

**Understand the VertiPaq engine**

Before jumping into optimization, it's important to understand what's happening under the hood of Power BI. By using compression algorithms and a multi-threaded query processor, the Analysis Services *VertiPaq engine* delivers fast access to tabular model objects and data by Power BI.

Power BI reads the content of your data source and transforms it into the internal VertiPaq columnar data structure, where each column is encoded and compressed. Dictionaries and indexes are created for each column. Finally, data structures are created for relationships and computation, and compression of calculated columns occur.

DAX queries are being processed by two engines, the *formula engine* and the *storage engine.* We'll talk more about these engines below.

**Describe DAX Studio**

[DAX Studio](https://daxstudio.org/) is an open-source tool for executing DAX queries against Power BI and Analysis Services models. DAX Studio is useful for:

* Reviewing the contents of your data model.
* Writing and optimizing complex DAX formulas and queries.

Download and install DAX Studio with the default settings, connect to your data model, and begin working on your queries. Once DAX Studio is installed, it can also be launched from the **External**Tools tab of the ribbon in Power BI desktop.

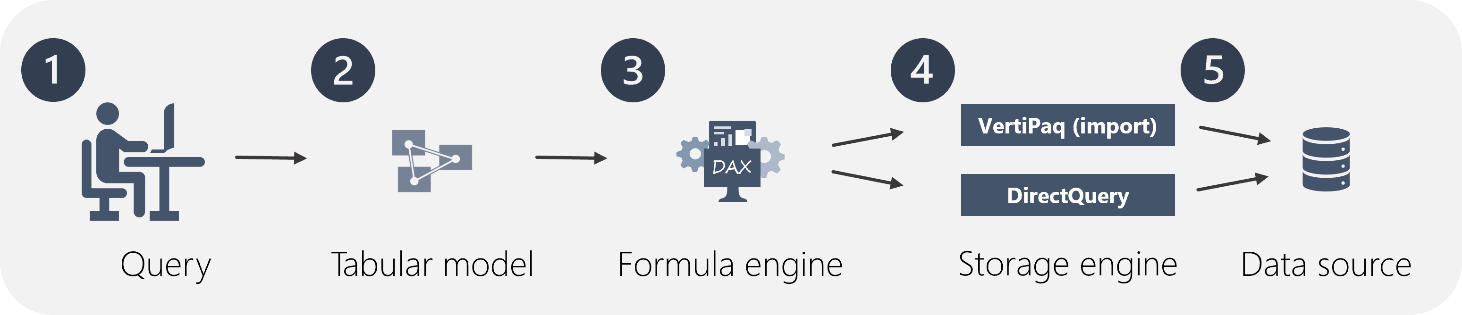
**Optimize the data model**

Now that you have an idea of how the VertiPaq engine works, let's discuss how you can use DAX Studio to optimize DAX queries that run in this environment.

Optimize DAX queries

Calculations using DAX, either measures or columns, are a part of a DAX query, which is processed by two engines in VertiPaq. When a query is processed, the *formula engine* processes the request, asks the storage engine for data, and performs necessary calculations. The *storage engine* retrieves and aggregates data requested by the formula engine.

In the diagram below, the DAX query is sent to the tabular model in steps 1 and 2. The request is then processed by the formula engine and sent to the storage engine, represented by step 3. In step 4, the storage engine either retrieves the data from the model and stores it in memory (for import mode), or passes the query on to the data source (for DirectQuery). For import mode, refreshing the data will retrieve the data from the source.

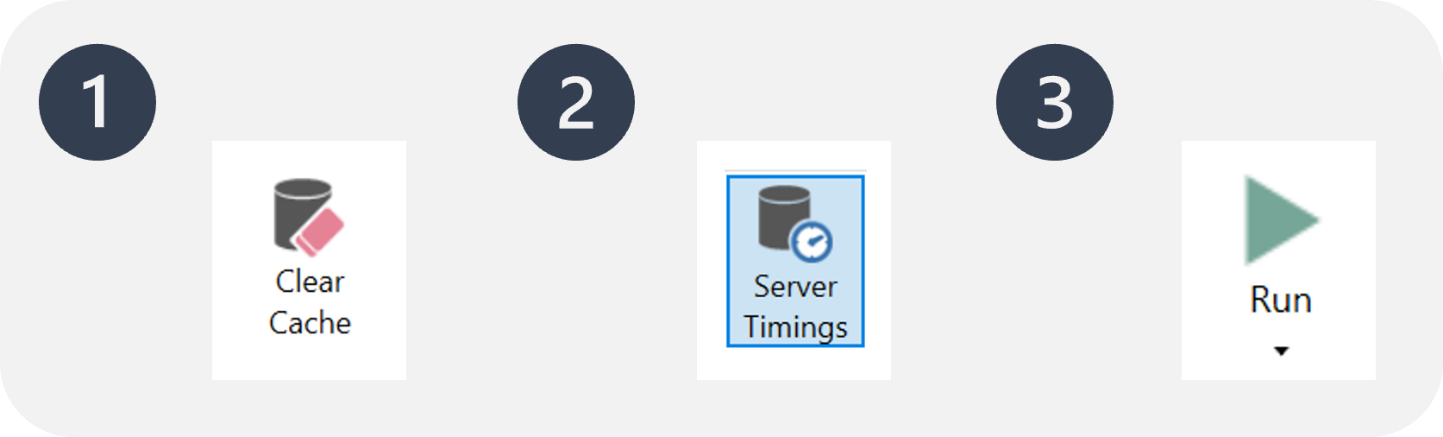


Troubleshooting in DAX Studio enables you to see detailed statistics on the server timings of your query. You're able to view the proportion of time the query takes in each engine and can then adjust your queries accordingly to improve performance.

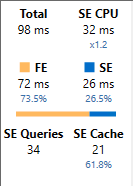
Let's talk through a scenario to understand how you can optimize a query using DAX Studio.

You have a report that contains a matrix visualizing 6 measures. Your CEO informs you that the visual is slow to render, and therefore the report is unusable. You start to dig in and confirmed slow render times using the Performance analyzer in Power BI desktop.

You then copy the query to look at it in DAX Studio to get more information on what might be causing your problem. In DAX Studio, you clear the cache (1), turn on the server timings (2), and then run the query (3).



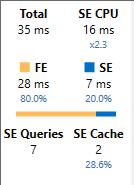
From top left to bottom right, the statistics tell you how many milliseconds it took to run the query, and the duration the storage engine (SE) CPU took. In this case, the formula engine (FE) took 73.5% of the time, while the SE took the remaining 26.5% of the time. There were 34 individual SE queries and 21 cache hits.



From here, you can investigate what in your measures might be causing the issues. This requires deep DAX knowledge and is sometimes a case of trial and error.

You experiment with the measures and recognize that improving the sales measure by replacing complicated IF statements in DAX with variables and a time intelligence function. A safe way to experiment is to comment on measures and rework them. You can comment on measures by typing two forward slashes at the beginning of a line (//). For multi-row comments, use /\* at the beginning of the comment and \*/ to close the comment.

After experimenting, you clear the cache and run the query again with the updates you made to the measure. You find that your updated measure performs much better, with nearly a 50% reduction in query execution time.



**Important**

The storage engine caches the results in memory for reuse. Because of this cache, it is critical to **clear the cache** prior to running queries in DAX Studio.

View model metrics using VertiPaq Analyzer

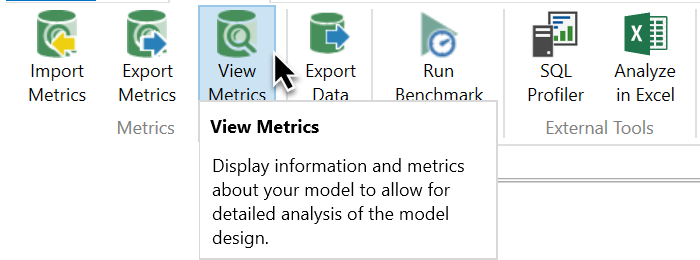
Viewing the *VertiPaq Analyzer Metrics* in DAX Studio is a great way to get an overall view of what's going on in your data model. VertiPaq Analyzer reports the memory consumption of the data model and can be used to quickly identify where you're spending the most memory. In short - you can use VertiPaq Analyzer to make memory gobbling offenders obvious, rectify them in Power BI, and then rerun VertiPaq Analyzer to see the immediate benefits of your data model updates.

**Note**

The VertiPaq engine only stores data in memory in *import* models. If you're using *DirectQuery*, the VertiPaq engine simply sends that query to the source. This means that viewing the VertiPaq Analyzer Metrics will only be helpful for *import* or *composite* models.

You can look at the size of the table, columns, etc., in bytes. The .pbix file further compresses these sizes - the displayed sizes in bytes are evaluated prior to compression.

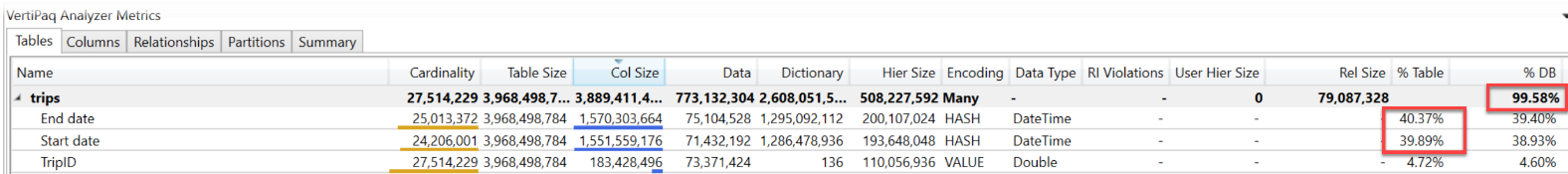
To view model metrics, launch DAX Studio from the external tools tab of the Power BI ribbon and select View Metrics from the Advanced tab in DAX Studio.



VertiPaq analyzer displays a number of important metrics about your model. We're going to focus specifically on memory consumption and cardinality. For a complete list of what each of these columns mean, consult the [DAX Studio documentation](https://daxstudio.org/docs/intro/).

Viewing metrics in DAX Studio helps you immediately find and fix problems. In this case, you can see that the problem is a column with high cardinality. You can then fix that issue back in Power BI, refresh the metrics, and immediately see the effects of your changes on the model.

For example, notice that the model in the image below contains a table that consumes 99.6% of the database memory. By drilling into the table, you can see that two columns, End date and Start date are gobbling up the most memory.



Take a look at those two columns back in Power BI desktop and notice that they're Date/time columns. Date/time columns inherently have high cardinality due to all of the possible combinations of dates and times.

A screenshot of a computer

AI-generated content may be incorrect.

Using the VertiPaq Analyzer in DAX Studio can help you easily identify and eliminate columns with high cardinality (including auto Date/time and floating-point decimal data types), and identify and remove columns that aren't used for anything.

**Note**

Refer to the [Power BI optimization guide](https://learn.microsoft.com/en-us/power-bi/guidance/power-bi-optimization#optimizing-the-data-model) for more detailed information on optimizing the data model.

# Optimize a data model by using Best Practice Analyzer

How can you be sure to implement data modeling best practices as you design and build data models in Power BI? There's a tool for that! The Best Practice Analyzer (BPA) in Tabular Editor can be used during the development of tabular models in Power BI or Analysis Services models.

**Describe Tabular Editor**

Tabular Editor is an alternative tool for authoring tabular models for Analysis Services and Power BI. Tabular Editor 2 is an open-source project that can edit a BIM file without accessing any data in the model. Tabular editor enables users to easily manipulate tabular models and can be used in a few different ways. Tabular Editor can be used to:

* quickly edit data models
* automate repetitive tasks using scripting
* incorporate DevOps with tabular models
* run BPA rules

This unit focuses on using Tabular Editor to run the BPA to ensure you're implementing data modeling best practices as you build.

**Describe the Best Practice Analyzer (BPA)**

BPA is a set of rules run in Tabular Editor that notify you of potential modeling missteps or changes that you can make to improve your model design and performance. It includes recommendations for naming, user experience, and common optimizations that you can apply to improve performance. For more information, see [Best practice rules to improve your model's performance](https://powerbi.microsoft.com/blog/best-practice-rules-to-improve-your-models-performance/).

BPA rules are a set of rules that you can add to your instance of Tabular Editor. When BPA rules are run on your tabular model, you'll get a list of rules that your model violates and can fix them using Tabular Editor.

BPA includes a [set of pre-defined rules](https://powerbi.microsoft.com/blog/best-practice-rules-to-improve-your-models-performance/), and you can also define your own rules to encourage certain conventions and practices when developing tabular models.

To use the pre-defined BPA rules, download the [BPA rules .json file](https://github.com/microsoft/Analysis-Services/tree/master/BestPracticeRules) from the GitHub repository. The rules are divided into categories for easier viewing, for example:

* Performance
* DAX Expressions
* Error Prevention
* Formatting
* Maintenance

Each rule has a description, and many of the rules also have a reference article/video. Reading the rule description and article will provide context as to why the rule is important and why you should follow it.

**Note**

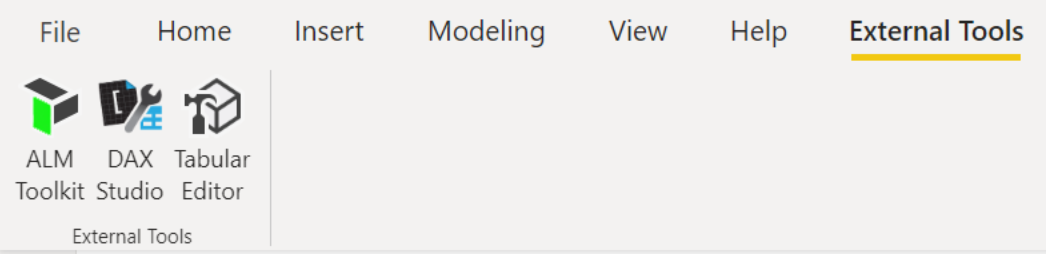
BPA rules can be run against tabular models in Power BI Desktop, SQL Server Analysis Services, Azure Analysis Services, or Power BI Premium.

**Run BPA in Tabular Editor**

1. Install [Tabular Editor](https://github.com/TabularEditor/TabularEditor).

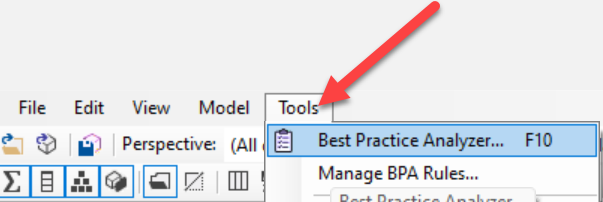
*Tabular Editor has an open-source and a paid version. This unit refers to use of the open-source version only.*

1. Download the Best Practice Rules from [GitHub](https://github.com/TabularEditor/BestPracticeRules/releases).
2. Save the file in **C:\Users\username\AppData\Local\TabularEditor** and name it **BPARules.json**.
3. In Power BI Desktop, select Tabular Editor on the External Tools tab of the ribbon.

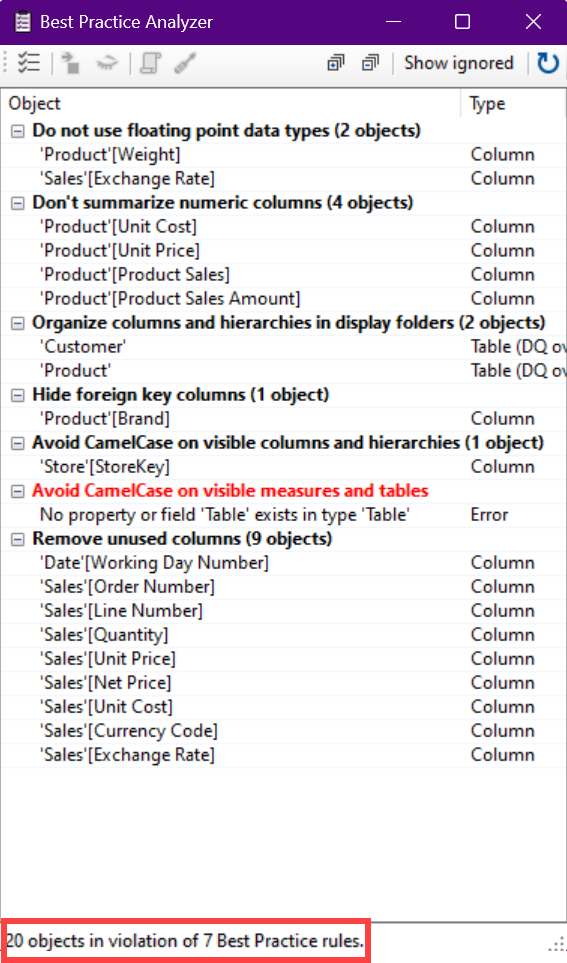


*Tabular Editor will automatically connect to the data model*

1. On the Tools tab, select **Best Practice Analyzer**.



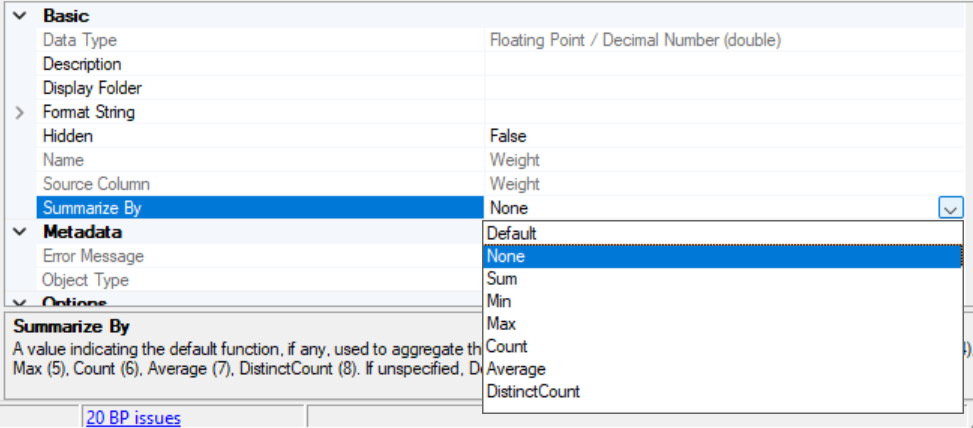
1. The Best Practice Analyzer window will open, displaying any violations of the Best Practice Rules.



**Note**

The best practice analyzer scans your model for issues whenever a change is made to the model. This is on by default but can be disabled.

1. Review the objects in violation of the rules and fix them using Tabular Editor. For example, let's fix the columns violating the **Don't summarize numeric columns** rule, starting with the Weight column in the Product table.
2. Double select (or right select) the object to go to the object in Tabular Editor. Change the object properties as necessary. In this case, we're changing **Summarize by** from **Sum**to **None**.



1. To save your changes to back to the Model.bim file, select **Save**, or use the keyboard shortcut **ctrl + s**.

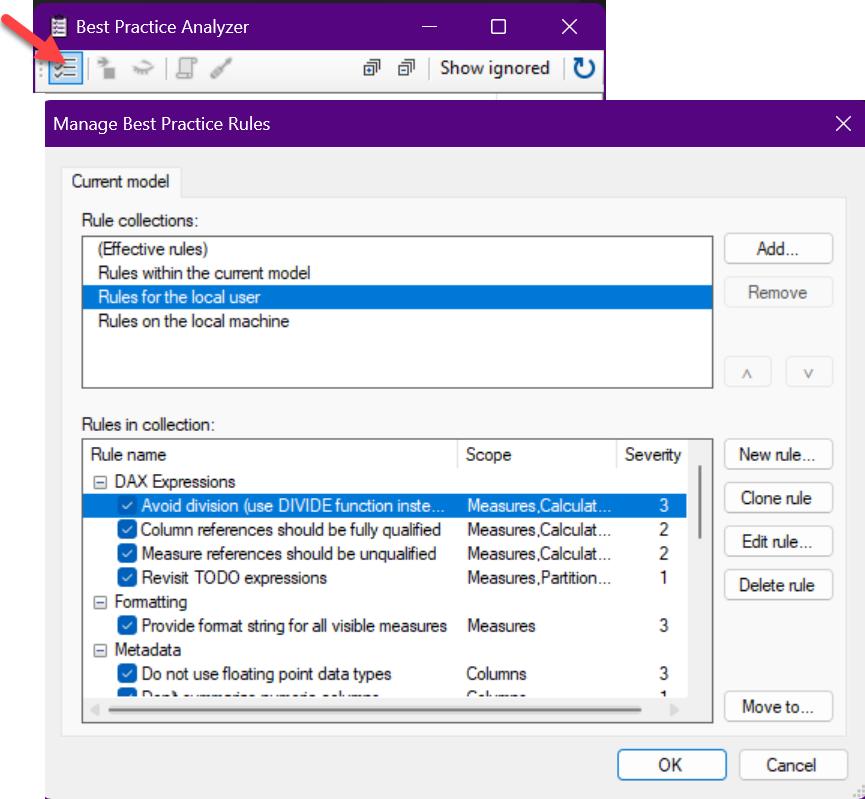
**Important**

Tabular Editor uses the [Tabular Object Model (TOM)](https://learn.microsoft.com/en-us/analysis-services/tom/introduction-to-the-tabular-object-model-tom-in-analysis-services-amo) to load and save metadata to and from Model.bim files. When fixing Best Practice rule violations, saving changes in Tabular Editor pushes modifications to the connected Power BI desktop data model.

**Customize BPA for your organization**

BPA was designed to enable you to create custom rules and best practices for your organization. If the Best Practice Rules don't suit your needs, you can create rules required for data modeling the best practices as you deem appropriate.

You can also edit existing rules, disable and ignore rules, and set severity levels for each rule. From the Best Practice Analyzer window, select **Manage Best Practice** **Rules**. This will enable you to turn rules to use in your scan on and off, edit rules, and delete rules. The edits you make to Best Practice Rules in this window will then be saved to the .json file.



Incorporate the use of BPA into your Continuous Integration/Continuous Deployment (CI/CD) process

If your organization has an established CI/CD process using Git, BPA can be integrated into your Azure Pipelines. If objects violate Best Practice Rules in the build, you can establish a process to either fail the build or proceed with warning based on the severity of the violation.

**Note**

The severity level that you set for each rule while managing your best practice rules only comes into play when deploying models using the [command line option](https://docs.tabulareditor.com/te2/Command-line-Options.html) in Tabular Editor.

Lab 2 – Controle de versão usando Github

# Introdução

Em ambientes colaborativos, onde múltiplos analistas e desenvolvedores trabalham simultaneamente em relatórios Power BI, a ausência de controle de versão pode gerar uma série de problemas:

* **Sobrescrita de arquivos**: É comum que alterações feitas por um membro da equipe sejam perdidas quando outro salva uma versão diferente do mesmo relatório.
* **Dificuldade de rastrear mudanças**: Sem histórico de versões, torna-se quase impossível identificar quem alterou o quê e quando.
* **Retrabalho e inconsistência**: A falta de sincronização entre versões pode levar à duplicação de esforços e à entrega de relatórios com dados ou visualizações inconsistentes.
* **Ausência de auditoria**: Sem versionamento, não há como garantir a integridade das alterações realizadas ao longo do tempo.

# Objetivos do aprendizado

Diante desses desafios, aprender a usar um versionador de código como o **GitHub** para controlar os arquivos .pbix e os artefatos relacionados aos relatórios Power BI se torna uma prática essencial. Os principais objetivos dessa abordagem incluem:

* ✅ **Rastreabilidade**: Cada alteração é registrada com autor, data e descrição, permitindo total transparência.
* ✅ **Colaboração segura**: Múltiplos usuários podem trabalhar em paralelo, com mecanismos para mesclar alterações e resolver conflitos.
* ✅ **Histórico de versões**: É possível retornar a versões anteriores do relatório com facilidade.
* ✅ **Automação e integração**: Com GitHub Actions, é possível automatizar testes, validações e até mesmo implantações de relatórios.
* ✅ **Padronização e governança**: O uso de branches, pull requests e revisões de código traz mais controle e qualidade ao processo de desenvolvimento dos relatórios.

# Resultado esperado

Ao finalizar este laboratório, você terá estabelecido um fluxo de atualização dos reportes de Power BI utilizando repositório Git do Github representado na imagem:

A diagram of a computer system

AI-generated content may be incorrect.

E terá alcançado os seguintes objetivos:

**✅ Integração entre Power BI Desktop e GitHub**

O estudante aprenderá a integrar o desenvolvimento de relatórios no Power BI com práticas modernas de controle de versão usando Git e GitHub.

**✅ Fluxo completo de versionamento**

O estudante será capaz de:

1. Abrir e gerenciar os arquivos no Visual Studio Code
2. Realizar commits e sincronizações com um repositório Git local
3. Enviar as alterações para uma branch de feature no GitHub
4. Criar um Pull Request (PR) para revisão
5. Mesclar as alterações na branch principal (main)

**Competências desenvolvidas**

* Uso de **Power BI Desktop** em ambientes versionados
* Manipulação de arquivos .pbix com **VS Code**
* Comandos básicos de **Git** (commit, sync, push)
* Colaboração e revisão de código via **GitHub Pull Requests**
* Organização de código em **branches** para ambientes de desenvolvimento

Esse fluxo é essencial para aplicar boas práticas de DevOps e BI moderno, especialmente em cenários corporativos com múltiplos desenvolvedores e ambientes de validação.

# Pré-requisitos

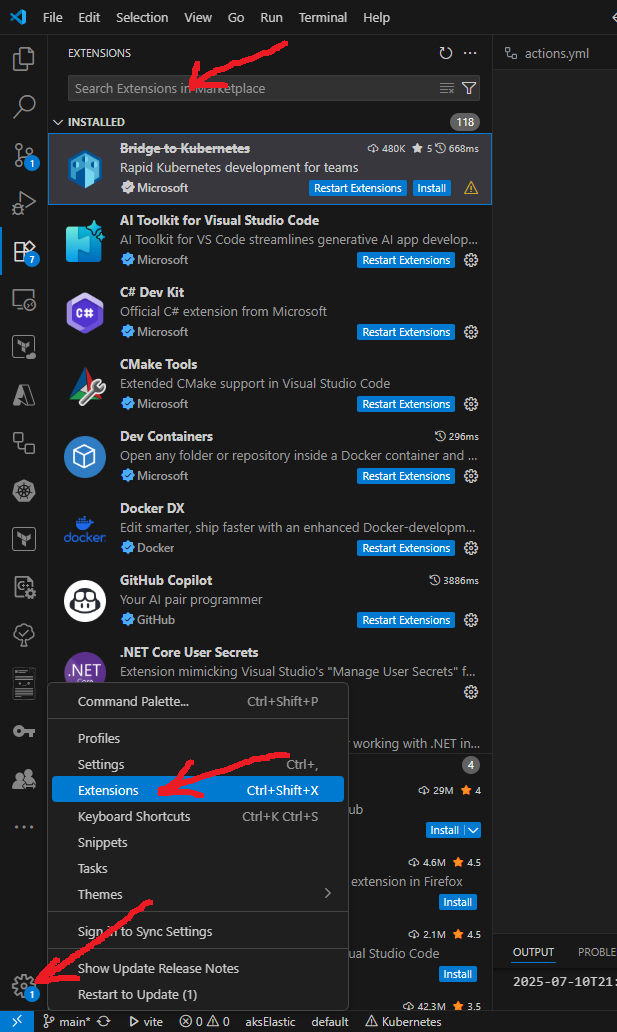
* Git 2.47.0 ou posterior instalado:

<https://git-scm.com/downloads>.

* Visual Studio Code instalado.

<https://code.visualstudio.com/download>

* Extensão C# para Visual Studio Code instalada.



**Visão Geral**

Neste laboratório, você aprenderá a:

* Clonar um repositório existente.
* Salvar trabalho com commits.
* Revisar histórico de mudanças.
* Trabalhar com branches usando Visual Studio Code.

# Instruções

**Passo 1: Configurar Git e Visual Studio Code**

* Abra o Visual Studio Code.
* No menu principal, selecione Terminal | New Terminal para abrir o painel TERMINAL.
* No TERMINAL, execute os seguintes comandos para configurar o Git:
  + 1. **git config --global credential.helper manager-core**

Este comando serve para configurar o Git para usar o **Git Credential Manager Core (GCM Core)** como gerenciador de credenciais. Esse é o método **mais moderno e seguro** para armazenar e gerenciar credenciais de acesso a repositórios remotos (como GitHub, Azure DevOps, GitLab etc.).

Explicando cada parte:

* git config: comando para configurar opções do Git.
* --global: aplica essa configuração para todos os repositórios do usuário.
* credential.helper: define o "ajudante" que vai gerenciar as credenciais.
* manager-core: O manager-core é uma evolução do antigo wincred e manager, compatível com **Windows, macOS, Linux.** Oferece suporte a Autenticação com **tokens de acesso pessoal (PATs),** Autenticação com **Oauth** e armazenamento seguro das credenciais (por exemplo, no Windows Credential Manager, no macOS Keychain ou no Linux Secret Service).

O que isso faz na prática?

Quando você clona, puxa ou empurra (push) para um repositório remoto que exige autenticação, o Git vai:

* Pedir seu usuário e senha (ou token).
* Salvar essas credenciais no gerenciador do Windows.
* Usar automaticamente essas credenciais nas próximas vezes, sem pedir novamente.
  + 1. **git config --global user.name "Seu Nome"**
    2. **git config --global user.email** [**seuemail@example.com**](mailto:seuemail@example.com)

**Passo 2: Clonar um Repositório**

* Acessar o github e logar-se: <https://github.com>
* No GitHub, crie um novo repositório ou use um repositório existente. A screenshot of a computer

  AI-generated content may be incorrect.
* Se for criar um, não esqueça de selecionar você como owner do repositório, se esta opção estiver disponível.

A screenshot of a computer

AI-generated content may be incorrect.

* Copie a URL do repositório.

Se for um repositório existente:

A screenshot of a computer

AI-generated content may be incorrect.

Se for um repositório novo:

A screenshot of a computer

AI-generated content may be incorrect.

* No Visual Studio Code, abra um novo terminal. Acesse a pasta, de preferência vazia, onde deseja clonar o repositório e execute o comando:

git clone <URL do repositório>

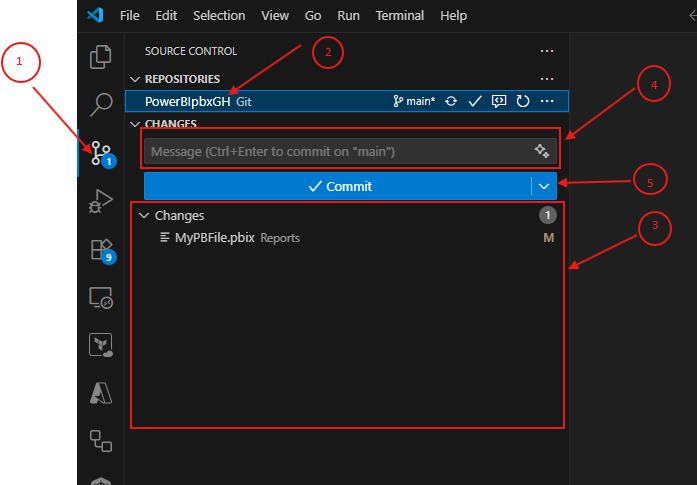
A screenshot of a computer program

AI-generated content may be incorrect.

Após a finalização da execução do comando, note que, na pasta onde o comando foi executado, foi criada uma nova pasta com o nome do repositório.

**Passo 3: Salvar Trabalho com Commits**

* **Fazer Commits**
  1. Crie uma pasta local Reports. Copie um arquivo pbix para dentro dessa pasta. Será o local onde você irá incluir os reportes de Power BI que passarão a versão controlado no GitHub. Altere normalmente os relatórios de Power BI e salve-o quando estiver satisfeito com as alterações.
  2. No Visual Studio Code
     + Vá para a aba SOURCE CONTROL.
     + Veja se está localizado na pasta onde está o diretório raíz do repositório.
     + Verifique quais os arquivos que foram alterados.
     + Adicione uma mensagem de commit que reflita as alterações executadas por você , pois esta mensagem será apresentada no histórico de alterações do seu repositório.
     + Pressione o botão commit . Esta ação irá atualizar o seu repositório local.



* 1. Clique no ícone de sincronização no canto inferior esquerdo para enviar as mudanças para o GitHub.

A screenshot of a computer

AI-generated content may be incorrect.

**Passo 4: Revisar Histórico**

* **Revisar Commits**
  1. No GitHub, navegue até a aba Commits do seu repositório para ver o histórico de commits.

A screenshot of a computer

AI-generated content may be incorrect.

**Passo 5: Trabalhar com Branches**

* **Criar Branches**
  + No Visual Studio Code, clique no nome do branch atual no canto inferior esquerdo, informe o nome da branch e selecione Create new branch.

A screenshot of a computer program

AI-generated content may be incorrect.

Note que o nome da branch agora aparece no canto inferior direito.

1. Adicione alguns arquivos novos em alguma pasta abaixo do diretório pelo Windows explorer.
2. Vá para a aba SOURCE CONTROL
3. Verifique quais os arquivos que foram alterados.
4. Adicione uma mensagem de commit que reflita as alterações executadas por você
5. Pessione o botão commit. Esta ação irá atualizar o seu repositório local. Note que um histórico vai ficando disponível à medida em que as alterações vão sendo *commitadas*.

A screenshot of a computer

AI-generated content may be incorrect.

**Passo 6: Trabalhar com Pull Requests**

* **Criar um Pull Request**
  1. Ao finalizar o passo 4, as alterações ainda estão em seu repositório local e o objetivo é atualizar o repositório remoto com as novas alterações, pois é lá onde o Pull Request será criado. Ainda aba SOURCE CONTROL, clique em Publish Branch. Este comando vai enviar a sua branch para o repositório.

A screenshot of a computer

AI-generated content may be incorrect.

* 1. No GitHub, no seu repositório, siga os passos de 1 a até 4 conforme mostrados na figura:

A screenshot of a computer

AI-generated content may be incorrect.

* + 1. Clique em <code>
    2. Clique no dropdown main
    3. Visualize sua branch e selecione-a. Verifique que os arquivos que você alterou no VS Code refletidos no repositório, mas apenas na branch.
    4. Clique em Compare & pull request.

Um **pull request (PR)** é uma forma de **propor mudanças** em um projeto que está hospedado em uma plataforma como o GitHub. Ele é muito usado em projetos colaborativos para permitir que outras pessoas revisem e aprovem alterações antes que elas sejam integradas ao código principal.

**Em termos simples, um pull request:**

* **Solicita que suas alterações sejam "puxadas" (pull)** para dentro da branch principal do projeto (geralmente chamada de main ou develop).
* **Permite revisão**: outras pessoas podem ver o que você mudou, comentar, sugerir melhorias ou aprovar.

Adicione as últimas informações para finalizar sua pull request conforme a figura abaixo:

A screenshot of a computer

AI-generated content may be incorrect.

* + 1. Título que ajuda a identificar do que se trata a pull request. Enriqueça com uma descrição, arquivos, cópias de tela, e informações que possam ajudar os revisores a entenderem o que foi feito e que as alterações estão de acordo.
    2. Você pode selecionar revisores. Neste caso, eles vão aprovar sua pull request
    3. Clique em Create pull request
* **Fazer o Merge & Pull**

A screenshot of a computer

AI-generated content may be incorrect.

Depois que você clica em **"Pull Request"** no GitHub, está iniciando um processo de **revisão e integração de código**.

Se você selecionou revisores, então eles verão o que **você mudou**. Eles podem **comentar**, **sugerir melhorias**, ou **fazer perguntas**.

Se tudo estiver certo, alguém com permissão (ou você mesmo, se tiver) pode **aprovar** o PR.

Depois disso, o código é **mesclado (merge)** com a branch principal.

Suas mudanças passam a fazer parte oficial do projeto.

* **Deletar a Branch**
  1. Clique em Delete branch conforme a figura abaixo.

A screenshot of a computer

AI-generated content may be incorrect.

* 1. No GitHub, no seu repositório, siga os passos de 1 a até 3 conforme mostrados na figura abaixo:

A screenshot of a computer

AI-generated content may be incorrect.

* + 1. Clique em <code>
    2. Clique no dropdown main
    3. Veja que a branch não existe mais no repositório. Veja que suas alterações estão na branch principal.

**Passo 7: Delete a branch do seu repositório local**

* Para deletar um branch, se ainda não tiver aberto um, abra um novo terminal e execute o comando:

git switch main >> para trocar de branch

git checkout -b <nome da branch>

A screenshot of a computer

AI-generated content may be incorrect.

**Passo 8: Atualize seu repositório local com o repositório remoto**

* Depois que você atualiza o repositório remoto no GitHub (por exemplo, com um merge de pull request ou alterações feitas diretamente lá), você pode **atualizar seu repositório local** para refletir essas mudanças utilizando o seguinte comando:

git pull origin nome-da-branch

Lab 3 – Pipeline de Deployment

# Introdução

Em muitos projetos de Business Intelligence, os relatórios Power BI são desenvolvidos, testados e publicados manualmente, o que pode gerar diversos desafios operacionais:

* **Publicações manuais e suscetíveis a erro**: A falta de padronização na publicação dos relatórios pode causar inconsistências entre ambientes de díficil rastreamento levando muito tempo para diagnosticar, corrigir e publicar um novo relatório.
* **Ambientes desalinhados**: É comum que o relatório em produção esteja diferente do que foi validado em teste, gerando retrabalho e perda de confiança.
* **Dificuldade de rastrear mudanças**: Sem automação, é difícil saber quando e por quem uma versão foi promovida para produção.
* **Baixa escalabilidade**: À medida que o número de relatórios cresce, manter o processo manual se torna insustentável.

# Objetivos do aprendizado

Para superar esses obstáculos, a implementação de uma **esteira de CI/CD** para os relatórios Power BI se torna uma solução estratégica. Essa abordagem permite que os arquivos .pbix sejam versionados, validados e promovidos automaticamente entre os ambientes de **Desenvolvimento**, **Teste** e **Produção**, com os seguintes objetivos:

* ✅ **Automatizar a publicação de relatórios**: Reduzindo erros humanos e garantindo consistência entre ambientes.
* ✅ **Padronizar o ciclo de vida dos relatórios**: Com regras claras de promoção e validação.
* ✅ **Melhorar a rastreabilidade**: Cada mudança é registrada e auditável. Assim fica fácil identificar a última versão estável, onde o problema ocorreu e agilidade na correção e publicação.
* ✅ **Facilitar rollback e controle de qualidade**: Com versionamento e testes automatizados.
* ✅ **Aumentar a agilidade da equipe**: Permite entregas mais rápidas e seguras.

# Resultado Esperado

Ao finalizar este laboratório, você terá desenvolvido um fluxo **implantação automatizada de relatórios do Power BI** utilizando práticas modernas de **DevOps com GitHub e Microsoft Fabric**, organizadas em um pipeline com três estágios: **Desenvolvimento, Teste e Produção**, conforme representado na imagem abaixo:

A close-up of a diagram

AI-generated content may be incorrect.

E terá alcançado os seguintes objetivos:

**✅ No GitHub**

* Acionar automaticamente o **GitHub Actions** para iniciar o processo de implantação

**✅ No pipeline de implantação do Power BI**

* Implantar os relatórios no workspace de **Desenvolvimento (pbi-dev)**
* Submeter as alterações para aprovação e promover para o workspace de **Teste (pbi-uat)**
* Após nova aprovação, promover para o workspace de **Produção (pbi-prd)**

**Competências desenvolvidas**

* Criação e uso de **GitHub Actions** para automação
* Configuração de **Deployment Pipelines no Power BI Fabric**
* Aplicação de boas práticas de **CI/CD para BI**

Esse fluxo garante que os relatórios sejam desenvolvidos, testados e implantados com controle, segurança e rastreabilidade.

# Pré-requisitos

* Lab 2
* Ter uma conta no Azure
* Ter uma capacidade para execução dos reportes Power BI, como Fabric ou Embedded.
* Capacidade em estado ativo

A screenshot of a computer

AI-generated content may be incorrect.

* A etiqueta de sensibilidade do arquivo deve ser configurada de acordo com a permissão da service principal. Se estiver protegida de modo que a service principal não tenha acesso, então não terá autorização para fazer o deployment no workspace.

A screenshot of a computer

AI-generated content may be incorrect.

# Instruções

Ferramentas como **GitHub Actions** e **Deployment pipeline do PowerBI/Fabric** podem ser integradas para criar pipelines que automatizam desde o commit do relatório até sua publicação nos workspaces corretos, com validações intermediárias e controle de acesso.

Adotar uma esteira de CI/CD para Power BI é um passo importante rumo à profissionalização da entrega de dados e insights, promovendo mais governança, eficiência e confiança nos processos analíticos da organização.

Neste laboratório nós vamos implementar um pipeline de deployment utilizando o **github actions** e também **Deployment pipeline do PowerBI/Fabric.**

**Passo 1: Criar uma app registration e um segredo para ela no Entra Id.**

Quando você quer que uma aplicação (como um pipeline de CI/CD no GitHub Actions) **execute ações no Azure** — como criar recursos, atualizar configurações ou fazer deploys — ela precisa de **autenticação**. Como não há um usuário humano envolvido, você precisa de uma **identidade de aplicação (**conhecida tambémcomo **Service Principal)**. No Azure, essa identidade é fornecida através de uma app registration e, para ela, uma secret associada.

**Por que usar App Registration com Secret?**

* **Identidade da aplicação**: O App Registration cria uma identidade única no Azure AD para a aplicação.
* **Autenticação segura**: O client secret funciona como uma "senha" que a aplicação usa para provar sua identidade.
* **Permissões controladas**: Você pode atribuir permissões específicas (via RBAC ou APIs) para que a aplicação só tenha acesso ao que realmente precisa.
* **Automação**: Permite que ferramentas como GitHub Actions façam deploys ou alterações sem intervenção manual.

Passo a passo: Criar uma App Registration e sua Secret

1. Acessar o Entra ID no Azure clicando no ícone abaixo:

A logo of a company

AI-generated content may be incorrect.

1. Criar uma nova Identidade da aplicação (app registration), seguindo os passos conforme mostrado na figura abaixo.

A screenshot of a computer

AI-generated content may be incorrect.

Na tela de cadastro, digite PBEmbedded como nome da identidade e clique em registrar

A screenshot of a computer

AI-generated content may be incorrect.

1. Gerar um “senha” para a aplicação (secret) conforme a sequência de telas abaixo:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Após clicar em adicionar (add), não se esqueça de copiar o valor da secret para ser armazenado como variável segura no Github do seu repositório.

**Nota**: Toda secret tem data de expiração e, portanto, deverá ser renovada de tempos em tempos.

A screenshot of a computer

AI-generated content may be incorrect.

**Passo 2: Criar os security groups**

Os **grupos de segurança** são usados para **gerenciar acesso a recursos de forma centralizada, segura e escalável**. Eles permitem que você atribua permissões a um grupo de identidades (usuários, aplicações ou service principals), em vez de configurar cada identidade individualmente.

Por que usar Security Groups?

* **Gerenciamento centralizado de permissões**  
  Em vez de atribuir permissões diretamente à App Registration, você atribui ao grupo. Isso facilita a manutenção e auditoria. Se em algum momento essa app registration não precisar mais desse acesso, basta remove-la do grupo.
* **Escalabilidade**  
  Se você tiver várias aplicações que precisam das mesmas permissões, basta adicioná-las ao grupo.
* **Segurança e conformidade**  
  Permite aplicar políticas de acesso e monitoramento ao grupo, mantendo o controle sobre quem tem acesso a quê.

Para o nosso laboratório, vamos criar 2 security groups:

* Um security group chamado **grp\_pbideploy** e, como membro desse grupo, vamos adicionar a Service Principal que criamos anteriormente. Será configurado como grupo como autorizado no Power BI Admin Portal. Assim, você garante que **apenas aplicações autorizadas** possam publicar ou acessar relatórios via API.
* Um security group chamado **grp\_pbiadmin** e, com membro desse grupo, vamos adicionar o seu usuário. Será configurado como grupo autorizado no Power BI Admin Portal para aprovar a sincronização de ambientes de Teste e Produção na esteira de CI/CD.

Criando o grupo para a App Registration (Service Principal)

Criar um **Security Group no Azure AD** e adicionar uma **App Registration** como membro tem como objetivo **centralizar e facilitar o controle de acesso** a recursos do Azure. Essa abordagem é especialmente útil em cenários de automação, como GitHub Actions, onde você quer aplicar permissões de forma gerenciável e segura.

Passo a passo: Criar Security Group e adicionar App Registration

1. Criar o grupo de segurança

* Acesse o portal do Azure: https://portal.azure.com
* Vá para **Azure Active Directory** > **Groups**
* Clique em **+ New group**
* Preencha os campos:
  + **Group type**: *Security*
  + **Group name**: algo como grp\_pbideploy
  + **Description**: opcional, mas útil para documentação
* Clique em **Create**

1. Adicionar a App Registration ao grupo

* Ainda dentro do grupo recém-criado, vá para a aba Members
* Clique em + Add members
* Na busca, digite o nome da App Registration criada no passo 2
* Selecione e clique em Select

Desta forma, as permissões a serem configuradas futuramente serão feitas para o grupo e valerão para todos os membros.

Criando o grupo para adicionar os usuários com permissão para sincronizar os ambientes de Teste e Produção.

Criar um **Security Group no Azure AD** e adicionar os usuários como membro tem como objetivo **centralizar e facilitar o controle de acesso** aos ambientes de relatórios, garantindo que apenas pessoas autorizadas possam realizar o deploy de relatórios.

Passo a passo: Criar Security Group e adicionar usuários

1. Criar o grupo de segurança

* Acesse o portal do Azure: https://portal.azure.com
* Vá para **Azure Active Directory** > **Groups**
* Clique em **+ New group**
* Preencha os campos:
  + **Group type**: *Security*
  + **Group name**: algo como grp\_pbiadmin
  + **Description**: opcional, mas útil para documentação
* Clique em **Create**

1. Adicionar seu usuário ao grupo

* Ainda dentro do grupo recém-criado, vá para a aba Members
* Clique em + Add members
* Na busca, digite o nome do seu usuário
* Selecione e clique em Select

Desta forma, as permissões a serem configuradas futuramente serão feitas para o grupo e valerão para todos os membros.

**Passo 3: Armazenar as credênciais da aplicação no GitHub.**

Agora que você já criou uma identidade para sua aplicação (app registration) e uma senha para ela (secret), você vai utilizar alguns dados gerados nesse processo para configurar as variáveis seguras AZURE\_CLIENT\_ID, AZURE\_TENANT\_ID e AZURE\_CLIENT\_SECRET no GitHub, que são requeridas para permitir que o GitHub Actions se autentique com segurança no Azure usando a identidade que foi criada com esse objetivo no passo anterior. Essas variáveis seguras são usadas para login automatizado via azure/login, permitindo que o pipeline execute ações como deploys, criação de recursos, etc.

1. Obter o valor a ser atribuído para cada uma das variáveis AZURE\_CLIENT\_ID, AZURE\_TENANT\_ID e AZURE\_CLIENT\_SECRET

Para as variáveis que vamos criar no Github, abaixo estão as instruções para preenchimento dos seus respectivos valores:

* **AZURE\_CLIENT\_ID**: Identificador único da aplicação registrada no Azure AD. Acesse a identidade da aplicação. A flecha vermelha indica o valor a ser utilizado nesta variável:   
  A screenshot of a computer

  AI-generated content may be incorrect.
* **AZURE\_TENANT\_ID**: Identificador do diretório (tenant) do Azure AD onde a aplicação está registrada. A flecha vermelha indica o valor a ser utilizado nesta variável:

A screenshot of a computer

AI-generated content may be incorrect.

* **AZURE\_CLIENT\_SECRET:** Senha gerada para a aplicação, usada para autenticação segura. O valor a ser utilizado nesta variável é o conteúdo da secret criada no Passo 1, item 3.

1. Configurar as variáveis seguras AZURE\_CLIENT\_ID, AZURE\_TENANT\_ID e AZURE\_CLIENT\_SECRET na interface web do Github.

* Acessar o github e logar-se: <https://github.com>
* Selecione o repositório que você criou no lab2:

A screenshot of a computer

AI-generated content may be incorrect.

* No menu superior, clique em "**Settings**" (Configurações).
* No menu lateral esquerdo, vá até "Secrets and variables" > "**Actions**".
* Clique em "**New repository secret**".
* Preencha os campos:
* Name: **AZURE\_CLIENT\_ID**
* Secret: cole o valor do **Client ID da App Registration no Azure**.
* Clique em "**Add secret**".
* Repita o processo para:
* **AZURE\_TENANT\_ID** (valor do **Tenant ID**)
* **AZURE\_CLIENT\_SECRET** (valor do client **secret** gerado no Azure)

**✅ Dica de segurança**

* Nunca exponha esses valores diretamente no código ou em logs.
* Use os secrets apenas dentro do contexto do GitHub Actions.

**Passo 4: Criar um GitHub Actions no VS Code**

O GitHub Actions é quem permite a automação de tarefas dentro do ciclo de vida de desenvolvimento de software, no caso, os relatórios de Power BI, diretamente no GitHub. Ele é uma plataforma de CI/CD (Integração Contínua e Entrega Contínua) que permite que você defina workflows que são executados automaticamente em resposta a eventos no repositório.

Principais Finalidades do GitHub Actions:

* Automatizar processos
* Build, testes, deploys, validações, etc.
* Exemplo: rodar testes automaticamente quando há um pull request.
* Facilitar o CI/CD
* Permite que você configure pipelines para compilar, testar e publicar seu código.
* Exemplo: fazer deploy automático para Azure, AWS, ou outro ambiente após um merge na branch main.
* Integrar com outras ferramentas
* Pode se conectar com serviços externos como Azure, Docker, Terraform, Slack, etc.
* Exemplo: enviar uma notificação para o Slack após um deploy.
* Executar scripts personalizados
* Você pode rodar qualquer comando ou script dentro dos workflows.
* Exemplo: rodar um script de migração de banco de dados após o deploy.
* Segurança e controle
* Usa secrets para proteger credenciais e tokens.
* Permite definir permissões específicas para cada job.

Como se faz isso na prática?

Você define um arquivo YAML dentro da pasta .github/workflows com os seguintes elementos:

* on: → define o gatilho (ex: push, pull\_request, schedule)
* jobs: → define os trabalhos que serão executados
* steps: → define os passos dentro de cada job (ex: checkout, login, build, deploy)

Vamos ao passo a passo para criar um Actions do Github usando VS Code

1. Abra o repositório no VS Code

2. Crie a pasta de workflows

* No VS Code, crie a pasta **.github/workflows**:
  + Clique com o botão direito no projeto > New Folder > nomeie como **.github**
  + Dentro dela, crie outra pasta chamada **workflows**

3. Crie o arquivo de workflow

* Dentro de .github/workflows, crie um arquivo chamado actions.yml (ou outro nome que preferir).
* Acesse a url abaixo e xole o conteúdo abaixo para dentro do arquivo yml:

<https://github.com/MartaMasson/PowerBIpbxGH/blob/main/.github/workflows/actions.yml>

4. Salve e faça commit

* Salve o arquivo.
* No terminal do VS Code:
* Siga os passos 5, 6, 7 e 8 do exercício anterior.

**Passo 5: Permitir o acesso às APIs do Power BI a partir da Service Principal que você criou anteriormente**

Ao usar **GitHub Actions** ou outras automações que interagem com o Power BI (via Fabric APIs), você precisa de uma forma segura e automatizada de autenticação. O **Service Principal** é uma identidade de aplicação registrada no Microsoft Entra ID (que você criou no Passo 1), que permite essa autenticação sem depender de credenciais de usuário.

No entanto, por padrão, o Power BI não permite que Service Principals acessem suas APIs. Para liberar esse acesso, você precisa:

* Habilitar a opção no **Admin Portal** do Power BI.
* **Restringir o acesso** a um grupo de segurança específico, do qual a Service Principal é membro, garantindo controle e segurança.

Passo a passo para habilitar o acesso no Power BI

1. Acesse https://app.powerbi.com com uma conta que tenha permissões de administrador.
2. No canto superior direito, clique em **Configurações (Settings)**.
3. Vá para o **Portal do Administrador (Admin Portal)**.
4. Dentro do Admin Portal, vá até a seção **Tenant settings (Configurações do locatário)**.
5. Role até encontrar **Developer settings (Configurações de desenvolvedor)**.
6. Localize a opção **"Service principals can use Fabric APIs"**.
7. Clique no botão de alternância para **habilitar** (deve ficar verde com "Enabled").
8. Em **Apply to**, selecione **"Specific security groups"**.
9. Adicione o grupo de segurança que contém o Service Principal (ex: grp\_pbideploy).
10. Clique em **Apply** para salvar as alterações.

A screenshot of a computer

AI-generated content may be incorrect.

**Passo 5: Criar o workspace de desenvolvimento no Power Bi**

Um workspace embedded permite que você incorpore relatórios do Power BI em aplicações externas (como portais, sistemas internos ou apps para clientes), usando autenticação via Service Principal ou usuário mestre. Isso é essencial para cenários de automação e integração com GitHub Actions, portais web, etc.

Você deve executar as instruções abaixo para criar o workspace de nome **pbi-dev**, que será o ambiente de desenvolvimento.

Passo-a-passo para criar um workspace no Power BI

1. Acesse https://app.powerbi.com
2. No menu lateral, clique em **Áreas de trabalho**
3. Clique em **+ Novo workspace**
4. Dê um nome de **pbi-dev** ao workspace e clique em **Aplicar**
5. Na parte superior da tela clique em **workspace settings**

A screenshot of a computer

AI-generated content may be incorrect.

1. Em **License Info**, configure a capacidade **Embedded** conforme a figura abaixo e feche a janela.

**A screenshot of a computer

AI-generated content may be incorrect.**

1. Em **Manage access**, clique em **Adicionar pessoas ou grupos** e inclua o grupo que você criou anteriormente do qual a Service Principal é membro. Então clique no botão **adicionar.** Depois de adicionar, configure-o com o papel de administrador

A screenshot of a phone

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Passo 6: Implantar os reportes no Power BI no Workspace pbi\_dev como o github actions com execução manual**

Agora vamos implantar os relatórios de Power BI no workspace pbi\_dev de forma automatizada.

1. Certificar que o workflow está configurado para permitir a execução manual

O workflow precisa estar configurado com o gatilho workflow\_dispatch, que permite execução manual. Para isso, verifique se o arquivo .github/workflows/actions.yml, inclui as seguintes instruções no início do arquivo:

on:

  workflow\_dispatch:

Esse gatilho habilita o botão de execução manual na interface do GitHub.

1. Vá para o repositório onde está o workflow.
2. Clique na aba **"Actions"** no topo da página.
3. Na lista de workflows à esquerda, clique no nome do workflow que deseja executar: **actions.yml**.
4. Clique no botão **"Run workflow"** no canto superior direito.
5. Clique em **"Run workflow"** novamente para iniciar.
6. Acompanhe a execução

* Após iniciar, o workflow aparecerá na lista de execuções.
* Clique nele para ver os logs em tempo real de cada etapa.

**Passo 7: Configurar o github actions para ser executado automaticamente quando se altera a branch main**

Para que um workflow do GitHub Actions seja executado **apenas quando houver alterações na branch main**, você precisa configurar o gatilho on.push corretamente no arquivo YAML do workflow.

1. Abra o arquivo do workflow no VS Code

Localize o arquivo dentro da pasta .github/workflows, por exemplo:  
actions.yml

1. Edite o gatilho do workflow

Substitua ou ajuste a seção on: para que fique assim:

on:

push:

branches:

- main

Esse trecho significa que o workflow será executado **somente** quando houver um push na branch main.

1. Salve, faça commit das alterações e sincronize com o repositório remoto

A partir de agora, o GitHub Actions só será disparado quando houver alterações (commits ou merges) diretamente na branch main.

Crie branches, altere algum reporte, faça o commit, pull requets e acompanhe a atualização dos reportes no workspace de forma automática (Laboratório 2).

**Passo 8: Criar os workspace de teste e produção no Power Bi**

Como parte da esteira de CI/CD, vamos criar os workspaces de teste e de produção.

Você deve executar as instruções abaixo para cada um dos seguintes workspaces com os seguintes nomes:

* **pbi-uat**, que será o ambiente de teste
* **pbi-prd**, que será o ambiente de produção

Passo-a-passo para criar um workspace no Power BI

1. Acesse <https://app.powerbi.com>
2. No menu lateral, clique em **Áreas de trabalho**
3. Clique em **+ Novo workspace**
4. Dê o nome ao workspace e clique em **Aplicar**
5. Na parte superior da tela clique em **workspace settings**

A screenshot of a computer

AI-generated content may be incorrect.

1. Em License Info, configure a capacidade Embedded conforme a figura abaixo e feche a janela.

**A screenshot of a computer

AI-generated content may be incorrect.**

1. Em Manage access, clique em Adicionar pessoas ou grupos e inclua o grupo **grp\_pbiadmin** que você criou anteriormente do qual o seu usuário é membro. Então clique no botão adicionar:

A screenshot of a phone

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Configurar o papel do grupo como administrador, da mesma forma que fez para o grp\_pbideploy no workspace de desenvolvimento.

A screenshot of a computer

AI-generated content may be incorrect.

**Passo 9: Criar o deployment pipeline no Power Bi**

Os **Deployment Pipelines** são uma funcionalidade do Power BI (agora integrado ao Microsoft Fabric) que permite **gerenciar o ciclo de vida de desenvolvimento de relatórios e dashboards** de forma estruturada e segura, usando **ambientes separados** como:

* **Desenvolvimento (Development)**
* **Teste (Test)**
* **Produção (Production)**

**A organização do ciclo de vida dos relatórios** permite que você desenvolva, teste e publique relatórios em ambientes distintos, evitando que alterações em desenvolvimento afetem diretamente os usuários finais.

**A automação e padronização do processo de publicação** permite que promova relatórios entre os ambientes com poucos cliques, mantendo consistência e controle.

**Facilidade de colaboração entre equipes:** Cada estágio pode ter permissões específicas, permitindo que desenvolvedores, analistas e administradores trabalhem de forma segura e coordenada.

**O Controle de acesso e segurança** integrado com o Entra ID permite o uso de **grupos de segurança do Entra ID** para definir quem pode administrar ou visualizar cada estágio do pipeline.

Passo a passo: Criar Deployment Pipeline no Power BI Fabric

1. Acesse o Power BI

* Vá para https://app.powerbi.com
* Faça login com sua conta corporativa

1. Criar o Deployment Pipeline

* No menu lateral esquerdo, clique em "**Pipelines de implantação**" (Deployment Pipelines)
* Clique em "+ **Criar pipeline**"
* Dê o nome: **Power BI Reports**
* Clique em "**Criar**"

1. Configurar os estágios do pipeline

**Estágio: Development**

* Clique em "**Conectar área de trabalho**" no estágio **Development**
* Selecione o workspace chamado **pbi-dev**
* Clique em "**Conectar**"

**Estágio: Test**

* Clique em "**Conectar área de trabalho**" no estágio **Test**
* Selecione o workspace chamado **pbi-uat**
* Clique em "**Conectar**"

**Estágio: Production**

* Clique em "**Conectar área de trabalho**" no estágio **Production**
* Selecione o workspace chamado **pbi-prd**
* Clique em "**Conectar**"

1. Gerenciar acesso aos workspaces

Para cada workspace (pbi-dev, pbi-uat, pbi-prd):

* Vá para **Configurações do workspace**
* Clique em **"Acesso"**
* Clique em **"+ Adicionar"**
* Adicione os grupos:
  + grp\_pbiadmin
* Defina o **nível de acesso como "Administrador"**
* Clique em **"Adicionar"**

1. Gerenciar acesso deployment pipeline

* Acesse o deployment pipeline **Power BI Reports**
* Vá para **Gerenciamento de Acesso (Manage access)**
* Clique em **"+ Adicionar pessoas ou grupos (Add people or groups)"**
* Adicione os grupos:
  + grp\_pbiadmin
* Defina o **nível de acesso como "Administrador (Admin)"**
* Clique em **"Adicionar"**

Você terá um pipeline chamado Power BI Reports com três estágios conectados aos respectivos workspaces. Os grupos grp\_pbideploy e grp\_pbiadmin terão acesso administrativo, permitindo automações e gerenciamento seguro.

A screenshot of a computer

AI-generated content may be incorrect.

**Passo 10: Sincronizar os reportes no Power Bi através do Deployment pipeline.**

A sincronização permite **promover relatórios, dashboards e datasets** de um ambiente para outro (ex: de Desenvolvimento para Teste, e de Teste para Produção), mantendo controle e consistência no ciclo de vida dos dados.

A screenshot of a computer

AI-generated content may be incorrect.

Passo a passo para sincronizar os estágios

1. Acesse o pipeline

* Vá para https://app.powerbi.com
* No menu lateral, clique em **"Pipelines de implantação"**
* Selecione o pipeline chamado **Power BI Reports**

1. Sincronizar de Development para Test

* No estágio **Development**, selecione os itens que deseja promover (relatórios, datasets, dashboards).
* Clique no botão **"Promover para Teste"** (ou "Deploy to Test").
* Aguarde a conclusão da operação.
* Verifique se os itens foram corretamente copiados para o workspace pbi-uat.

A screenshot of a computer

AI-generated content may be incorrect.

1. Sincronizar de Test para Production

* No estágio **Test**, selecione os itens que deseja promover.
* Clique em **"Promover para Produção"** (ou "Deploy to Production").
* Aguarde a conclusão da operação.
* Confirme que os itens estão disponíveis no workspace pbi-prd.

**A screenshot of a computer

AI-generated content may be incorrect.**

**Boas práticas**

* **Verifique dependências**: certifique-se de que os datasets e relatórios estão funcionando corretamente antes de promover.
* **Use grupos de segurança**: como já configurado, os grupos grp\_pbideploy e grp\_pbiadmin devem ter acesso de administrador para realizar essas ações.
* **Documente alterações**: mantenha um registro das versões promovidas para facilitar auditoria e rollback, se necessário.