

# Power BI developer solutions documentation

Solutions and best practices for Power BI developers of all kinds. It provides a suite of features that enable developer focused capabilities like external tool support, script and API support, source control with Git integration, and Continuous Integration/Continuous Delivery (CI\CD) with Azure DevOps. Microsoft Power BI Desktop developer mode brings Pro BI developer experiences right into Power BI Desktop.

## Power BI Desktop projects

### CONCEPT

[Power BI Desktop projects overview](#)

[Project semantic model folder](#)

[Project report folder](#)

[Project Git integration](#)

[Project Azure DevOps integration](#)

## Automation with Power BI REST APIs

### CONCEPT

[Using the Fabric REST APIs](#)

[Using the Power BI REST APIs](#)

[Interact with Power BI reports using the Power BI REST API](#)

## Power BI CI/CD

### CONCEPT

[Introduction to deployment pipelines](#)

[Azure DevOps Integration](#)

[Continuous integration with Azure DevOps and Git](#)

## Solutions and Best Practices

### {CONCEPT}

[Hot and cold table partitions to optimize very large Power BI data models](#)

[External tools in Power BI Tabular Object Model \(TOM\)](#)

[Programming Power BI semantic models with the Tabular Object Model \(TOM\)](#)

## External tools

### {CONCEPT}

[External tools in Power BI Desktop](#)

[Register an external tool](#)

# Power BI Desktop projects (PREVIEW)

Article • 01/20/2025

## ⓘ Important

Power BI Desktop projects is currently in preview.

## ⓘ Tip

For guidance about how to plan a Power BI development, see [Power BI implementation planning](#).

Power BI Desktop introduces a new way to author, collaborate, and save your projects. When you save your work as a **Power BI Project** (PBIP), report and semantic model *item* definitions are saved as individual plain text files in a simple, intuitive folder structure.

Saving your work as a project has the following benefits:

- **Text editor support** - Item definition files are formatted text files containing semantic model and report metadata. These files are publicly documented and human readable. While project files support simple text editing tools like Notepad, it's better to use a code editor like [Visual Studio Code \(VS Code\)](#) ↗, which provides a rich editing experience including intellisense, validation, and Git integration.
- **Programmatic generation and editing item definitions** - You can programmatically generate and modify item definition text files, enabling batch operations such as updating all report pages visuals or adding a set of measures to each table. For semantic models, you can use the [Tabular Object Model \(TOM\)](#) client library to deserialize the semantic model metadata, make programmatic modifications, and serialize it back to the files.
- **Source control** - Power BI semantic model and report item definitions can be stored in a source control system, like Git. With Git, you can track version history, compare revisions (diff), and revert to previous versions. Source control can also unblock collaboration when using Power BI Desktop by using familiar collaboration mechanisms for resolving conflicts (merge) and reviewing changes (pull requests). To learn more, see [Version control in Git](#).
- **Continuous Integration and Continuous Delivery (CI/CD)** - You can use systems where developers in your organization submit a proposed change to the CI/CD

system. The system then validates the change with a series of *quality gates* before applying the change to the production system. These quality gates can include code reviews by other developers, automated testing, and automated build to validate the integrity of the changes. CI/CD systems are typically built on top of existing source control systems. To learn more, see [DevOps - Continuous integration](#), and [DevOps - Continuous delivery](#).

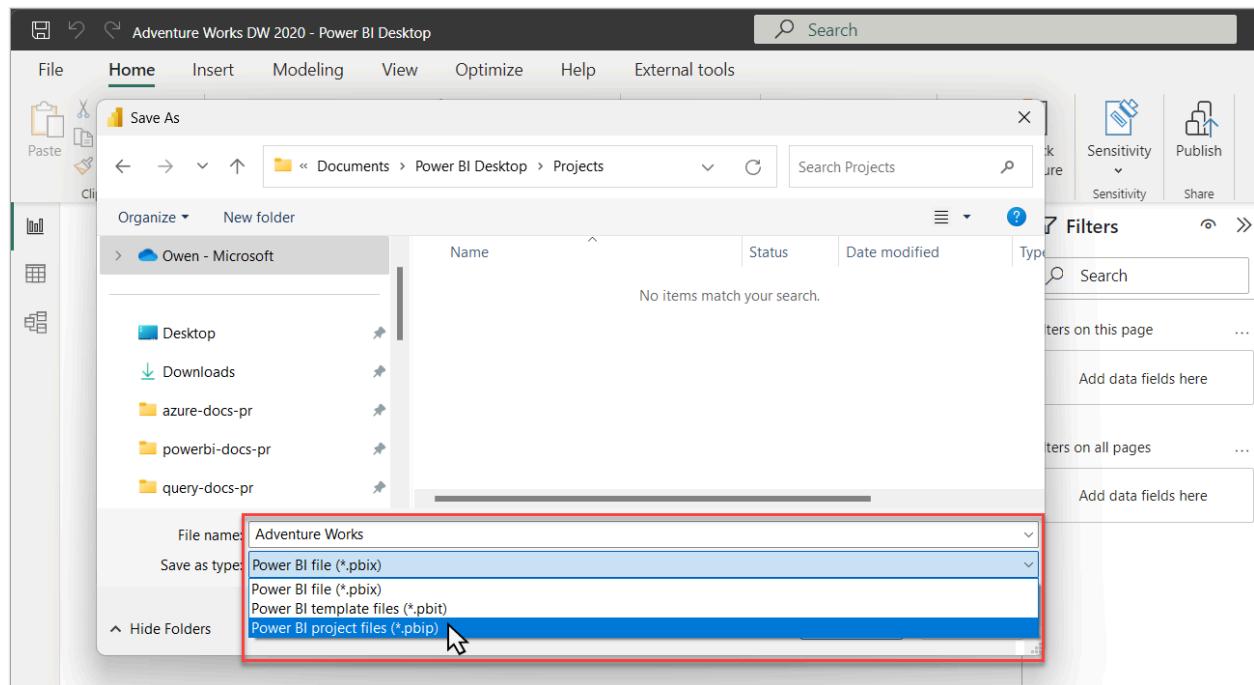
## Enable preview features

Saving as a Power BI Project in Power BI Desktop is currently in **preview**, and you must enable it in **Preview features**.

Go to **File > Options and settings > Options > Preview features** and check the box next to **Power BI Project (.pbip) save option**.

## Save as a project

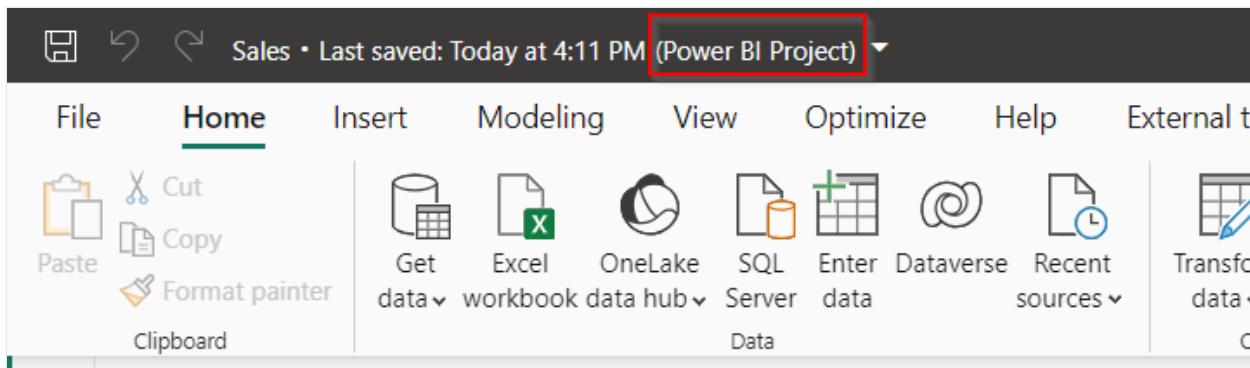
If you're working on a new project or you opened an existing Power BI Desktop file (pbix), you can save your work as a Power BI *project* file (pbip):



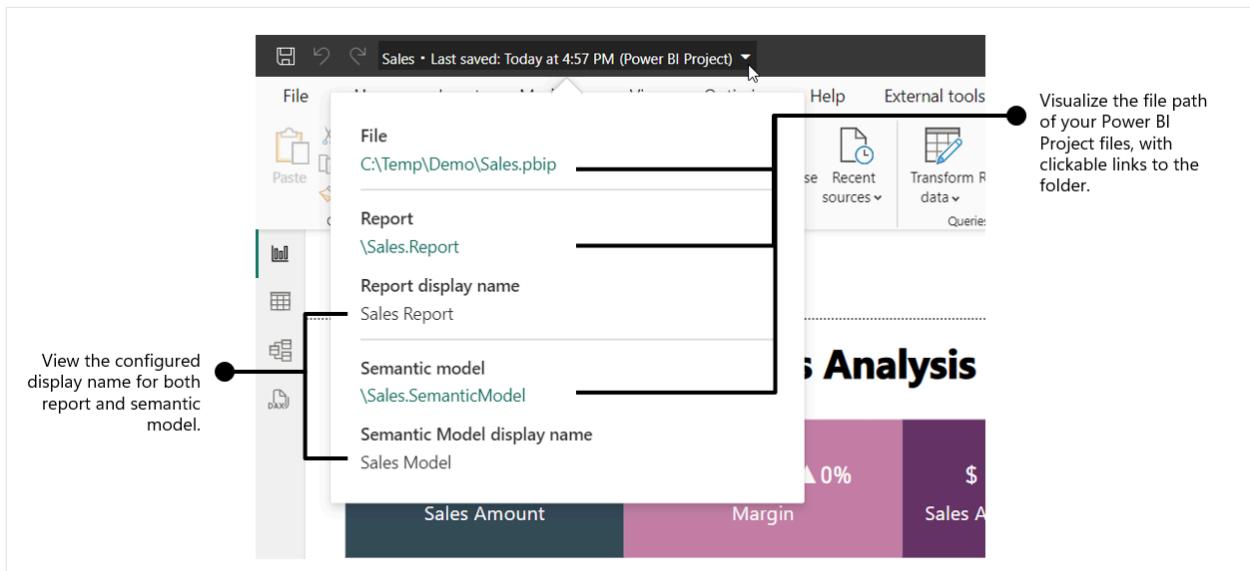
When you save as a project, Power BI Desktop saves report and semantic model items as folders, each containing text files that define the item:

Name	Date modified
AdventureWorks.Report	19/03/2024 20:43
AdventureWorks.SemanticModel	19/03/2024 20:43
.gitignore	15/03/2024 10:43
AdventureWorks.pbip	18/03/2024 12:21

After saving as a project, you can see when you're working on a project by looking at the title bar:



If you select on the title bar, a flyout appears that's specific for Power BI Project. This flyout lets you locate the project files and the display name settings for the report and the semantic model. You can also open the folder in file explorer by clicking on the paths.



Let's take a closer look at what you see in your project's root folder:

## <project name>.SemanticModel

A collection of files and folders that represent a Power BI semantic model. To learn more about the files and subfolders and files in here, see [Project Semantic Model folder](#).

## <project name>.Report

A collection of files and folders that represent a Power BI report. To learn more about the files and subfolders and files in here, see [Project report folder](#).

## .gitignore

Specifies intentionally untracked files Git should ignore for Power BI Project files, such as the cache.abf and localSettings.json.

Power BI Desktop creates the [.gitignore](#) file only if one doesn't already exist in the chosen save folder or parent Git repository.

Default content of .gitignore when saving as PBIP:

```
**/.pbip/localSettings.json  
**/.pbip/cache.abf
```

## <project name>.pbip

The PBIP file contains a pointer to a report folder, opening a PBIP opens the targeted report and model for authoring.

For more information, see the [pbip schema document](#).

# Open a Power BI Project

You can open Power BI Desktop from the Power BI Project folder either by opening the **pbip** file or the **pbir** file in the report folder. Both options open the report for editing, and the semantic model, if there's a relative reference to a semantic model.

You can save multiple reports and semantic models to the same folder. Having a separate pbip file for each report isn't required because you can open each report directly from the **.pbir** within the report folder.

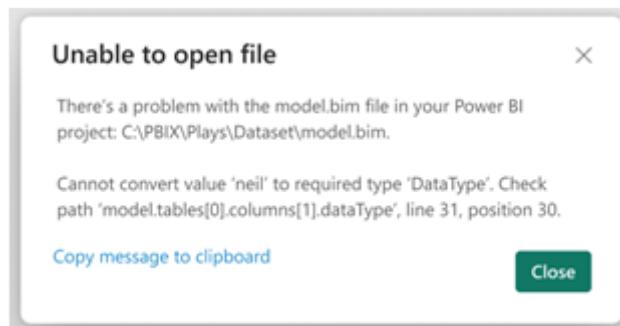
```
project
  AdventureWorks-Sales.Report
    definition.pbir
  AdventureWorks-Stocks.Report
    definition.pbir
  AdventureWorks.SemanticModel
    ...
.gitignore
AdventureWorks.pbip
```

## Changes outside Power BI Desktop

When saved as a project, you're not forced into making changes to your semantic model and report definitions only in Power BI Desktop. You can use other tools such as VS Code, open-source community tools like Tabular Editor, or even Notepad. However, not every file or change supports editing by external, open-source tools.

Changes to files or properties outside of Power BI Desktop can cause unexpected errors, or even prevent Power BI Desktop from opening. In those cases, you must resolve the issues in the files before trying to open the project again in Power BI Desktop.

If possible, Power BI Desktop indicates the file and location of error:



Schema details for the following files aren't documented. During preview, changes to these files outside of Power BI Desktop aren't supported:

- Report\
  - [report.json](#)
  - [mobileState.json](#)
  - [semanticModelDiagramLayout.json](#)
- SemanticModel\
  - [diagramLayout.json](#)

## Deploy to Fabric workspace

When working with Power BI project files, you can deploy your content to a Fabric workspace using the following publishing mechanisms:

- Use [Fabric Git Integration](#).
- Use [Fabric APIs](#).
- Use [Power BI Desktop publish option](#).

### ⓘ Note

Publishing through [Power BI Desktop publish](#) uses a temporary PBIX file that is published to the service, similar to saving and publishing a PBIX file. Unlike other PBIP deployment options that only deploy metadata, this publishing method deploys both the metadata and the [local data cache](#) of the semantic model being edited.

## Model authoring

You can make changes to the semantic model definition by using external tools in two ways:

- By connecting to Power BI Desktop's Analysis Service (AS) instance with [external tools](#).
- By editing JSON metadata in the model.bim file using VS Code or another external tool.

Not every model object supports write operations. Applying changes outside of those supported can cause unexpected results.

Objects that support write operations:

[+] [Expand table](#)

Object	Connect to AS instance	File change / TMDL view
Tables	No	Yes
Columns	Yes <sup>1, 2</sup>	Yes
Calculated tables	Yes	Yes
Calculated columns	Yes	Yes
Hierarchies	Yes	Yes

Object	Connect to AS instance	File change / TMDL view
Relationships	Yes	Yes
Measures	Yes	Yes
Model KPIs	Yes	Yes
Calculation groups	Yes	Yes
Perspectives	Yes	Yes
Translations	Yes	Yes
Row Level Security (RLS)	Yes	Yes
Object Level Security (OLS)	Yes	Yes
Annotations	Yes	Yes
M expressions	No	Yes <sup>3, 4</sup>

Keep in mind:

- Any changes to open files made outside Power BI Desktop requires a restart for those changes to be shown in Power BI Desktop. Power BI Desktop isn't aware of changes to project files made by other tools.
- Power BI Desktop doesn't support tables with multiple partitions. Only a single partition for each table is supported. Creating tables with empty partitions or more than one partition results in an error when opening the report.
- Automatic date tables created by Power BI Desktop shouldn't be changed by using external tools.
- When changing a model that uses Direct Query to connect a Power BI semantic model or Analysis Services model, you must update the `ChangedProperties` and `PBI_RemovedChildren` collection for the changed object to include any modified or removed properties. If `ChangedProperties` and/or `PBI_RemovedChildren` isn't updated, Power BI Desktop might overwrite any changes the next time the query is edited or the model is refreshed in Power BI Desktop.
- 1 - Changing a column's data type is supported. However, renaming columns isn't supported when connecting to the AS instance.
- 2 - If the semantic model has the [Auto date/time](#) feature enabled, and you create a new datetime column outside of Power BI Desktop, the local date table isn't automatically generated.

- 3 - Partition [SourceType](#) must be Calculated, M, Entity, or CalculationGroup. Partition [Mode](#) must be Import, DirectQuery, or Dual.
- 4 - Any expression edits outside of Power BI Desktop in a project with [unappliedChanges.json](#) are lost when those changes are applied.
- Modifying table query expressions outside of Power BI Desktop results in the removal of the table data upon restarting Power BI Desktop.

## JSON file schemas

Most project files contain metadata in JSON format. Corresponding JSON schemas can be used for validation and documentation.

With JSON schemas, you can:

- Learn about configurable properties.
- Use inline JSON validation provided by the code editor.
- Improve authoring with syntax highlighting, tooltips, and autocomplete.
- Use external tools with knowledge of supported properties within project metadata.

Use VS Code to map JSON schemas to the files being authored. JSON schemas for project files are provided in the [Power BI Desktop samples Git repo ↗](#).

## Considerations and limitations

- Power BI Desktop isn't aware of changes made with other tools or applications. Changes made by using external tools require you to restart Power BI Desktop before those changes are shown.
- Sensitivity labels aren't supported with Power BI projects.
- Diagram view is ignored when editing models in the Service.
- When saving as a Power BI Project, the maximum length of the project files path is 260 characters.
- In Power BI Desktop, you can't save as a PBIP directly to OneDrive and SharePoint.
- When editing PBIP files outside of Power BI Desktop, they should be saved using UTF-8 without BOM encoding.
- Report Linguistic Schema isn't supported with Power BI projects.
- Power BI Desktop uses CRLF as end-of-line. To avoid problems in your diffs, configure Git to handle line endings by enabling [autocrlf ↗](#).
- Power BI Projects is currently not supported in Microsoft Power BI Desktop version optimized for Power BI Report Server.

- Live connect reports saved as PBIP require the [XMLA Endpoint](#) to be enabled.

## Frequently asked questions

**Question:** Looking at semantic model and report item folder definitions only a few files are marked as required, what happens if I delete them?

**Answer:** Power BI Desktop automatically creates them when you save as a project (PBIP).

**Question:** Is Power BI Desktop aware of changes I make to the Power BI Project files from an external tool or application?

**Answer:** No. Any change made to the files requires Power BI Desktop to be restarted to reflect those changes.

**Question:** If I convert a PBIX to a PBIP, can I convert it back to PBIX?

**Answer:** Yes. You can save a PBIX as a PBIP, or save a PBIP as a PBIX.

**Question:** Can I convert PBIX into PBIP and vice-versa programmatically?

**Answer:** No. You can only convert a PBIX into a PBIP and vice-versa using Power BI Desktop's **File > Save as**.

**Question:** Can I deploy a Power BI Desktop project to Azure Analysis Services (AAS) or SQL Server Analysis Services (SSAS)?

**Answer:** No. Power BI Desktop project report definitions aren't supported in AAS and SSAS. And model definitions use an enhanced metadata unique to Power BI. For AAS and SSAS projects, use Microsoft Visual Studio for model authoring, Git, and Azure DevOps integration.

**Question:** Why isn't there a \*.pbip file when I connect my Fabric workspace to Git? How can I edit my report and semantic model in Power BI Desktop?

**Answer:** The PBIP file is optional and simply serves as a shortcut to the report folder. You can open both the report and the semantic model for editing in Power BI Desktop by opening the definition.pbir file located in the report folder.

## Related content

- [Power BI Desktop project semantic model folder](#)
- [Power BI Desktop project report folder](#)
- [Power BI Desktop projects Git integration](#)

- Power BI Desktop projects Azure DevOps integration
  - External tools in Power BI Desktop
- 

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Ask the community ↗](#)

# Power BI Desktop project semantic model folder

Article • 08/13/2024

## ⓘ Important

Power BI Desktop projects is currently in preview.

This article describes the files and subfolders in a Microsoft Power BI Desktop project's **Semantic Model** folder. The files and subfolders here represent a Power BI semantic model. Depending on your project, the semantic model folder can include:

- .pbil\
  - localSettings.json
  - editorSettings.json
  - cache.abf
  - unappliedChanges.json
- definition.pbism<sup>1</sup>
- model.bim<sup>2</sup>
- definition\ folder<sup>3</sup>
- diagramLayout.json
- .platform

<sup>1</sup> - This file is required.

<sup>2</sup> - This file is required when saving using TMSL format.

<sup>3</sup> - This file is required when saving using TMDL format.

Not every project semantic model folder includes all of the files and subfolders described here.

## Semantic Model files

### .pbil\localSettings.json

Contains semantic model settings that apply only for the current user and computer. It should be included in gitlgnore or other source control exclusions. By default, Git ignores this file.

For more information, see the [localSettings.json schema document](#).

## .pbileditorSettings.json

Contains semantic model editor settings saved as part of the semantic model definition for use across users and environments.

For more information, see the [editorSettings.json schema document](#).

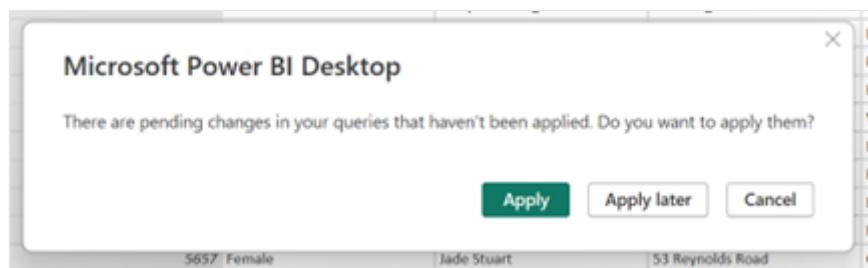
## .pbilcache.abf

An Analysis Services Backup (ABF) file containing a local cached copy of the model and data when it was last edited. It should be included in gitignore or other source control exclusions. By default, Git ignores this file.

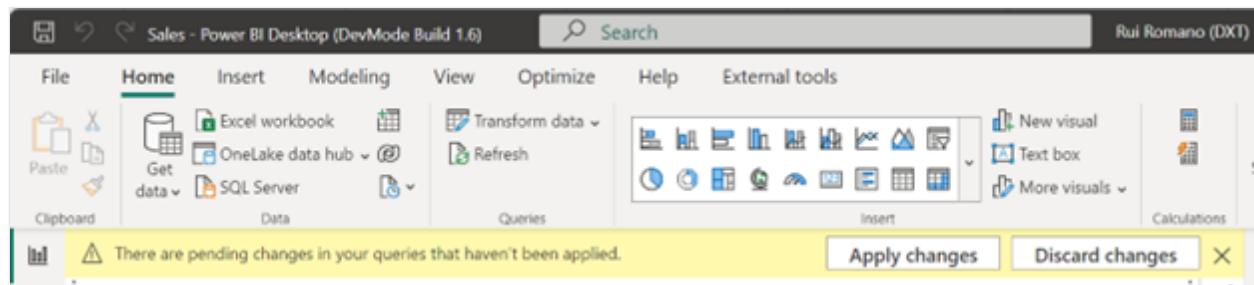
Power BI Desktop can open a project without a cache.abf file. In that case, it opens the report connected to a model with its entire definition but without data. If a cache.abf exists, Power BI Desktop loads the data and overwrites the model definition with the content in model.bim.

## .pbilunappliedChanges.json

Power BI Desktop allows you to save changes made in the Transform Data editor (Power Query) without first applying those changes to the data model.



When you select **Apply later**, the unapplied changes are saved into the unappliedChanges.json file. When pending changes are in the unappliedChanges file, Power BI Desktop prompts you to apply or discard those pending changes:



If you select **Apply changes**, Power BI Desktop overwrites the queries in model.bim with the queries from unappliedChanges.json. If you edited queries in model.bim outside of Power BI Desktop and there's a previous unappliedChanges.json file, your changes are

lost and replaced by the queries in `unappliedChanges.json` when those changes are applied.

The `unappliedChanges.json` file is automatically incorporated into the semantic model definition and saved in Git by default. This allows you to commit your ongoing work to the development branch, serving as a backup and making it accessible to other team members. However, you can exclude this file from Git's tracking, preventing unfinished query work from affecting other developers.

For more information, see the [unappliedChanges.json schema document](#).

## definition.pbism

Contains the overall definition of a semantic model and core settings.

This file also specifies the supported semantic model definition formats through the 'version' property.

[ ] [Expand table](#)

Version	Supported formats
1.0	Semantic model definition must be stored as TMSL in the <code>model.bim</code> file.
4.0 or above	Semantic model definition can be stored as TMSL ( <code>model.bim</code> file) or TMDL (\definition folder).

For more information, see the [definition.pbism schema document](#).

## model.bim

This file is only available if the Power BI project is saved using the TMSL format. It contains a Tabular Model Scripting Language (TMSL) [Database object](#) definition of the project model.

## definition\ folder

This folder is only available if the Power BI project is saved using the [TMDL format](#). It replaces the `model.bim` file.

This folder contains a [Tabular Model Definition Language \(TMDL\) Database object](#) definition of the project model.

## diagramLayout.json

Contains diagram metadata that defines the structure of the semantic model associated with the report. During **PREVIEW**, this file doesn't support external editing.

## .platform

Fabric platform file that holds properties vital for establishing and maintaining the connection between Fabric items and Git.

To learn more, see [Git integration automatically generated system files](#).

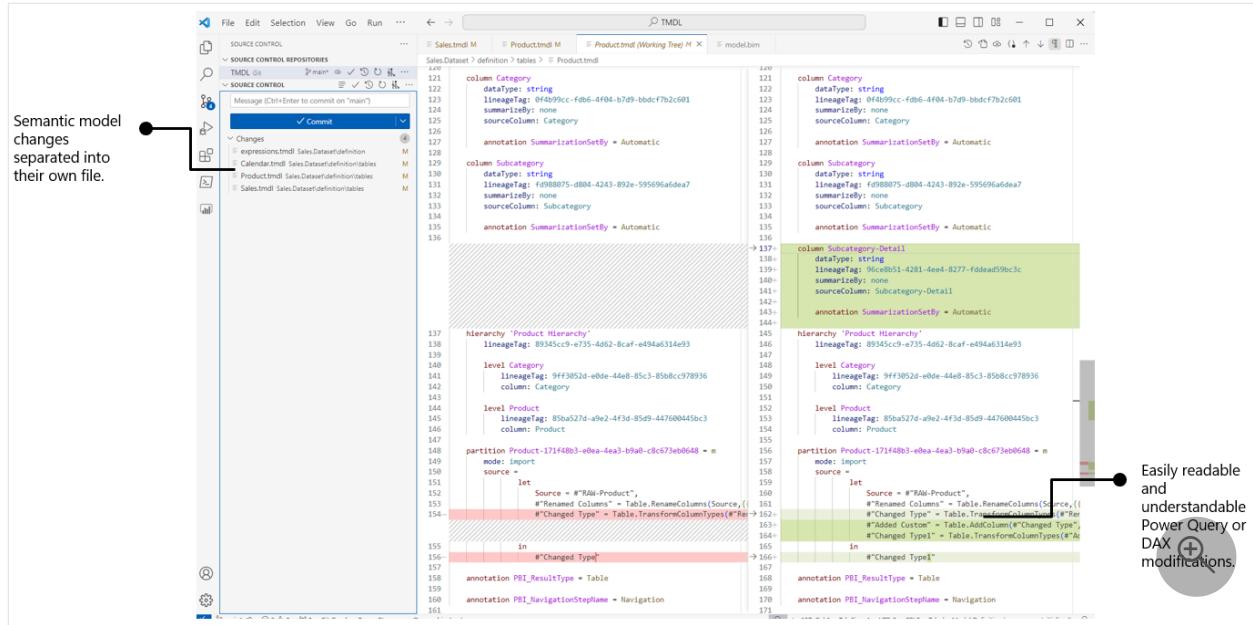
## TMDL format

With the objective of providing a better source control and co-development experience, you can save your Power BI Project files (PBIP) using Tabular Model Definition Language (TMDL) as the semantic model file format.

Unlike Tabular Model Scripting Language (TMSL), TMDL has been designed from the ground up to be human-friendly, facilitating not only *readability* but also easy *editing* in any text editor. This represents a substantial enhancement for source control and *collaborative development* experiences.

<b>TMSL</b>	<b>TMDL</b>
<pre>14590 14591 14592 14593 14594 14595 14596 14597 14598 14599 14600 14601 14602 14603 14604 14605 14606 14607 14608 14609 14610 14611 14612 14613 &gt; 14614 14615 14616 14617 14618 14619 14620 14621 14622 14623 14624 14625 14626 14627 14628 14629 14630 14631 14632 14633 14634 14635 14636 14637 14638 14639 14640 14641 14642 &gt; 14643 14644 14645 14646 14647 14648 14649 14650 14651 14652 14653 14654 14655 14656 14657 14658 14659 14660 14661 14662 14663 14664 14665 14666 14667 14668 14669 14670 14671 14672 14673 14674 14675 14676 14677 14678 14679 14680 14681 14682 14683 14684 14685 14686 14687 14688 14689 14690 14691 14692 14693 14694 14695 14696 14697 14698 14699 14700 14701 14702 14703 14704 14705 14706 14707 14708 14709 14710 14711 14712 14713 14714 14715 14716 14717 14718 14719 14720 14721 14722 14723 14724 14725 14726 14727 14728 14729 14730 14731 14732 14733 14734 14735 14736 14737 14738 14739 14740 14741 14742 14743 14744 14745 14746 14747 14748 14749 14750 14751 14752 14753 14754 14755 14756 14757 14758 14759 14760 14761 14762 14763 14764 14765 14766 14767 14768 14769 14770 14771 14772 14773 14774 14775 14776 14777 14778 14779 14780 14781 14782 14783 14784 14785 14786 14787 14788 14789 14790 14791 14792 14793 14794 14795 14796 14797 14798 14799 14800 14801 14802 14803 14804 14805 14806 14807 14808 14809 14810 14811 14812 14813 14814 14815 14816 14817 14818 14819 14820 14821 14822 14823 14824 14825 14826 14827 14828 14829 14830 14831 14832 14833 14834 14835 14836 14837 14838 14839 14840 14841 14842 14843 14844 14845 14846 14847 14848 14849 14850 14851 14852 14853 14854 14855 14856 14857 14858 14859 14860 14861 14862 14863 14864 14865 14866 14867 14868 14869 14870 14871 14872 14873 14874 14875 14876 14877 14878 14879 14880 14881 14882 14883 14884 14885 14886 14887 14888 14889 14890 14891 14892 14893 14894 14895 14896 14897 14898 14899 14900 14901 14902 14903 14904 14905 14906 14907 14908 14909 14910 14911 14912 14913 14914 14915 14916 14917 14918 14919 14920 14921 14922 14923 14924 14925 14926 14927 14928 14929 14930 14931 14932 14933 14934 14935 14936 14937 14938 14939 14940 14941 14942 14943 14944 14945 14946 14947 14948 14949 14950 14951 14952 14953 14954 14955 14956 14957 14958 14959 14960 14961 14962 14963 14964 14965 14966 14967 14968 14969 14970 14971 14972 14973 14974 14975 14976 14977 14978 14979 14980 14981 14982 14983 14984 14985 14986 14987 14988 14989 14990 14991 14992 14993 14994 14995 14996 14997 14998 14999 14999 15000 15001 15002 15003 15004 15005 15006 15007 15008 15009 150010 150011 150012 150013 150014 150015 150016 150017 150018 150019 150020 150021 150022 150023 150024 150025 150026 150027 150028 150029 150030 150031 150032 150033 150034 150035 150036 150037 150038 150039 150040 150041 150042 150043 150044 150045 150046 150047 150048 150049 150050 150051 150052 150053 150054 150055 150056 150057 150058 150059 150060 150061 150062 150063 150064 150065 150066 150067 150068 150069 150070 150071 150072 150073 150074 150075 150076 150077 150078 150079 150080 150081 150082 150083 150084 150085 150086 150087 150088 150089 150090 150091 150092 150093 150094 150095 150096 150097 150098 150099 150099 150100 150101 150102 150103 150104 150105 150106 150107 150108 150109 150110 150111 150112 150113 150114 150115 150116 150117 150118 150119 150120 150121 150122 150123 150124 150125 150126 150127 150128 150129 150130 150131 150132 150133 150134 150135 150136 150137 150138 150139 150140 150141 150142 150143 150144 150145 150146 150147 150148 150149 150150 150151 150152 150153 150154 150155 150156 150157 150158 150159 150160 150161 150162 150163 150164 150165 150166 150167 150168 150169 150170 150171 150172 150173 150174 150175 150176 150177 150178 150179 150180 150181 150182 150183 150184 150185 150186 150187 150188 150189 150189 150190 150191 150192 150193 150194 150195 150196 150197 150198 150199 150199 150200 150201 150202 150203 150204 150205 150206 150207 150208 150209 150210 150211 150212 150213 150214 150215 150216 150217 150218 150219 150219 150220 150221 150222 150223 150224 150225 150226 150227 150228 150229 150229 150230 150231 150232 150233 150234 150235 150236 150237 150238 150239 150239 150240 150241 150242 150243 150244 150245 150246 150247 150248 150249 150249 150250 150251 150252 150253 150254 150255 150256 150257 150258 150259 150259 150260 150261 150262 150263 150264 150265 150266 150267 150268 150269 150269 150270 150271 150272 150273 150274 150275 150276 150277 150278 150279 150279 150280 150281 150282 150283 150284 150285 150286 150287 150288 150289 150289 150290 150291 150292 150293 150294 150295 150296 150297 150298 150299 150299 150300 150301 150302 150303 150304 150305 150306 150307 150308 150309 150309 150310 150311 150312 150313 150314 150315 150316 150317 150318 150319 150319 150320 150321 150322 150323 150324 150325 150326 150327 150328 150329 150329 150330 150331 150332 150333 150334 150335 150336 150337 150338 150339 150339 150340 150341 150342 150343 150344 150345 150346 150347 150348 150349 150349 150350 150351 150352 150353 150354 150355 150356 150357 150358 150359 150359 150360 150361 150362 150363 150364 150365 150366 150367 150368 150369 150369 150370 150371 150372 150373 150374 150375 150376 150377 150378 150379 150379 150380 150381 150382 150383 150384 150385 150386 150387 150388 150389 150389 150390 150391 150392 150393 150394 150395 150396 150397 150398 150399 150399 150400 150401 150402 150403 150404 150405 150406 150407 150408 150409 150409 150410 150411 150412 150413 150414 150415 150416 150417 150418 150419 150419 150420 150421 150422 150423 150424 150425 150426 150427 150428 150429 150429 150430 150431 150432 150433 150434 150435 150436 150437 150438 150439 150439 150440 150441 150442 150443 150444 150445 150446 150447 150448 150449 150449 150450 150451 150452 150453 150454 150455 150456 150457 150458 150459 150459 150460 150461 150462 150463 150464 150465 150466 150467 150468 150469 150469 150470 150471 150472 150473 150474 150475 150476 150477 150478 150479 150479 150480 150481 150482 150483 150484 150485 150486 150487 150488 150489 150489 150490 150491 150492 150493 150494 150495 150496 150497 150498 150499 150499 150500 150501 150502 150503 150504 150505 150506 150507 150508 150509 150509 150510 150511 150512 150513 150514 150515 150516 150517 150518 150519 150519 150520 150521 150522 150523 150524 150525 150526 150527 150528 150529 150529 150530 150531 150532 150533 150534 150535 150536 150537 150538 150539 150539 150540 150541 150542 150543 150544 150545 150546 150547 150548 150549 150549 150550 150551 150552 150553 150554 150555 150556 150557 150558 150559 150559 150560 150561 150562 150563 150564 150565 150566 150567 150568 150569 150569 150570 150571 150572 150573 150574 150575 150576 150577 150578 150579 150579 150580 150581 150582 150583 150584 150585 150586 150587 150588 150589 150589 150590 150591 150592 150593 150594 150595 150596 150597 150598 150599 150599 150600 150601 150602 150603 150604 150605 150606 150607 150608 150609 150609 150610 150611 150612 150613 150614 150615 150616 150617 150618 150619 150619 150620 150621 150622 150623 150624 150625 150626 150627 150628 150629 150629 150630 150631 150632 150633 150634 150635 150636 150637 150638 150639 150639 150640 150641 150642 150643 150644 150645 150646 150647 150648 150649 150649 150650 150651 150652 150653 150654 150655 150656 150657 150658 150659 150659 150660 150661 150662 150663 150664 150665 150666 150667 150668 150669 150669 150670 150671 150672 150673 150674 150675 150676 150677 150678 150679 150679 150680 150681 150682 150683 150684 150685 150686 150687 150688 150689 150689 150690 150691 150692 150693 150694 150695 150696 150697 150698 150699 150699 150700 150701 150702 150703 150704 150705 150706 150707 150708 150709 150709 150710 150711 150712 150713 150714 150715 150716 150717 150718 150719 150719 150720 150721 150722 150723 150724 150725 150726 150727 150728 150729 150729 150730 150731 150732 150733 150734 150735 150736 150737 150738 150739 150739 150740 150741 150742 150743 150744 150745 150746 150747 150748 150749 150749 150750 150751 150752 150753 150754 150755 150756 150757 150758 150759 150759 150760 150761 150762 150763 150764 150765 150766 150767 150768 150769 150769 150770 150771 150772 150773 150774 150775 150776 150777 150778 150779 150779 150780 150781 150782 150783 150784 150785 150786 150787 150788 150789 150789 150790 150791 150792 150793 150794 150795 150796 150797 150798 150799 150799 150800 150801 150802 150803 150804 150805 150806 150807 150808 150809 150809 150810 150811 150812 150813 150814 150815 150816 150817 150818 150819 150819 150820 150821 150822 150823 150824 150825 150826 150827 150828 150829 150829 150830 150831 150832 150833 150834 150835 150836 150837 150838 150839 150839 150840 150841 150842 150843 150844 150845 150846 150847 150848 150849 150849 150850 150851 150852 150853 150854 150855 150856 150857 150858 150859 150859 150860 150861 150862 150863 150864 150865 150866 150867 150868 150869 150869 150870 150871 150872 150873 150874 150875 150876 150877 150878 150879 150879 150880 150881 150882 150883 150884 150885 150886 150887 150888 150889 150889 150890 150891 150892 150893 150894 150895 150896 150897 150898 150899 150899 150900 150901 150902 150903 150904 150905 150906 150907 150908 150909 150909 150910 150911 150912 150913 150914 150915 150916 150917 150918 150919 150919 150920 150921 150922 150923 150924 150925 150926 150927 150928 150929 150929 150930 150931 150932 150933 150934 150935 150936 150937 150938 150939 150939 150940 150941 150942 150943 150944 150945 150946 150947 150948 150949 150949 150950 150951 150952 150953 150954 150955 150956 150957 150958 150959 150959 150960 150961 150962 150963 150964 150965 150966 150967 150968 150969 150969 150970 150971 150972 150973 150974 150975 150976 150977 150978 150979 150979 150980 150981 150982 150983 150984 150985 150986 150987 150988 150989 150989 150990 150991 150992 150993 150994 150995 150996 150997 150998 150998 150999 150999 151000 151001 151002 151003 151004 151005 151006 151007 151008 151009 151009 151010 151011 151012 151013 151014 151015 151016 151017 151018 151019 151019 151020 151021 151022 151023 151024 151025 151026 151027 151028 151029 151029 151030 151031 151032 151033 151034 151035 151036 151037 151038 151039 151039 151040 151041 151042 151043 151044 151045 151046 151047 151048 151049 151049 151050 151051 151052 151053 151054 151055 151056 151057 151058 151059 151059 151060 151061 151062 151063 151064 151065 151066 151</pre>	

just looking at the folder and files. Ultimately, this leads to a great source control and co-development experience when dealing with git diff's and merge conflicts.



The screenshot shows the Power BI TMDL editor interface. On the left, there is a sidebar titled "SOURCE CONTROL" with a tree view showing "Changes" and "Sales.tmdl". A callout bubble points to this area with the text "Semantic model changes separated into their own file." In the center, the main editor window displays the TMDL code for the semantic model. A callout bubble points to the right side of the code with the text "Easily readable and understandable Power Query or DAX modifications." The code itself is a large block of JSON-like syntax, showing definitions for tables, columns, and partitions, including lineage tags and annotations for Power BI features like navigation and PBI\_ResultType.

```

{
    "Sales": {
        "Dataset": {
            "definition": "table > Product"
        }
    }
}

```

Learn more about TMDL [here](#).

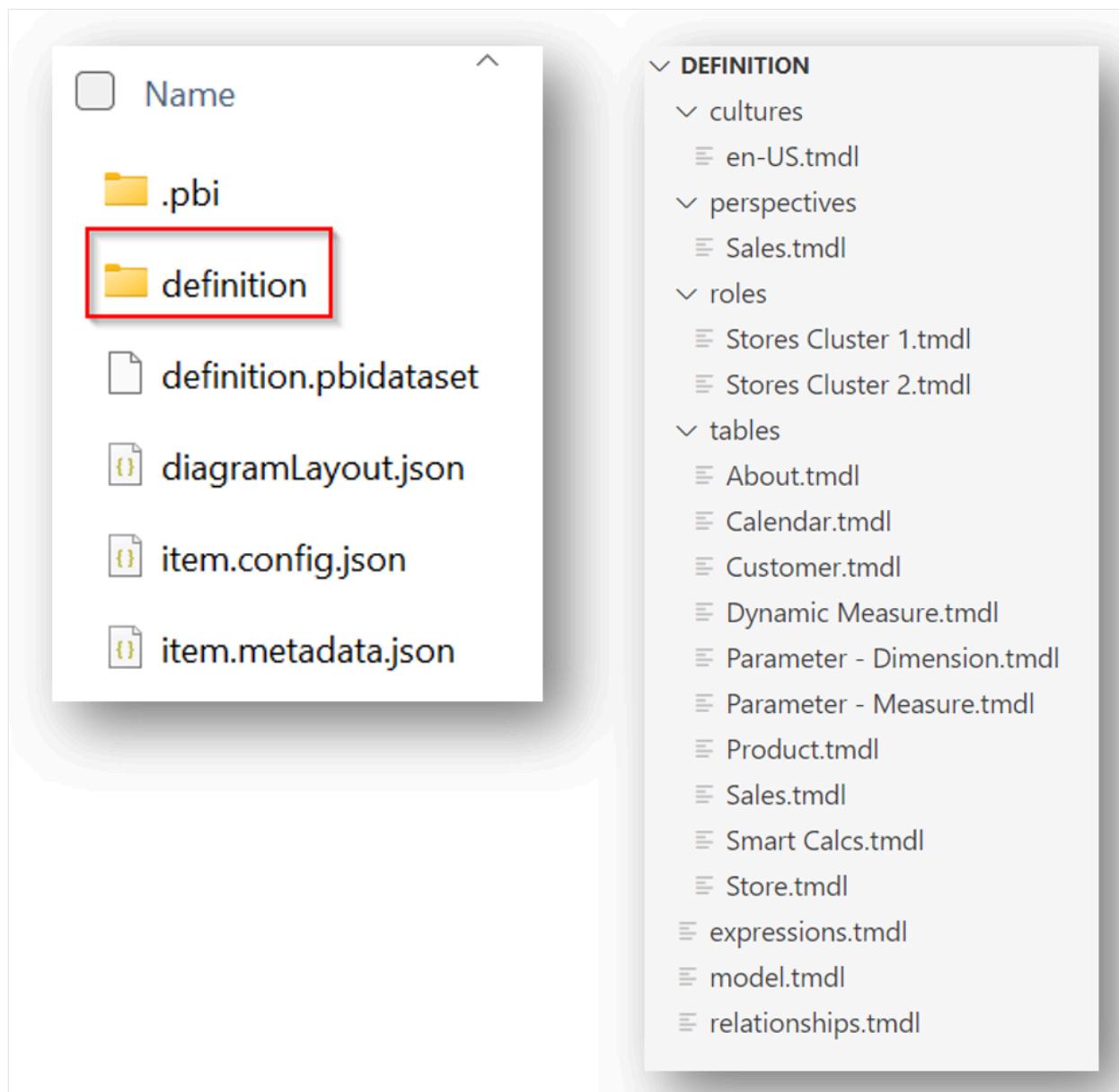
## Enable TMDL format Preview feature

Saving as a Power BI Project using TMDL is currently in preview. Before using it, you must first enable it in Preview features:

Go to File > Options and settings > Options > Preview features and check the box next to **Store semantic model using TMDL format**.

## Save as a project using TMDL

With the TMDL Preview feature enabled, when you save a project, your semantic model is saved as a TMDL folder named `\definition` inside of [semantic model folder](#):



Learn more about the [TMDL folder structure](#).

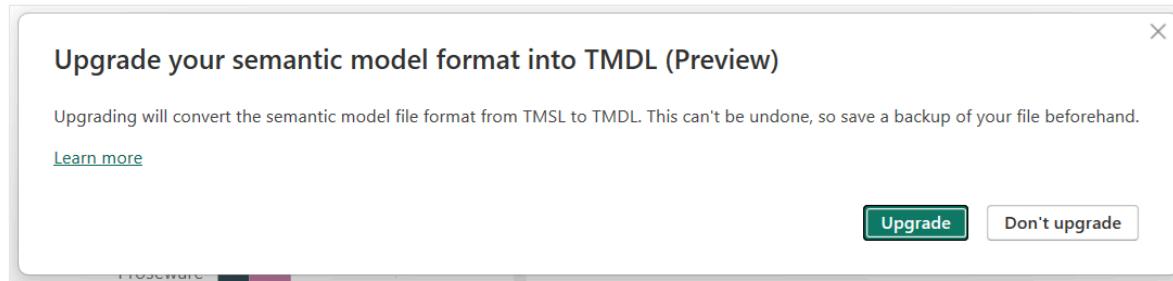
## Convert existing PBIP to TMDL

If you already have a PBIP using TMSL as semantic model format, you can convert it to TMDL as follows:

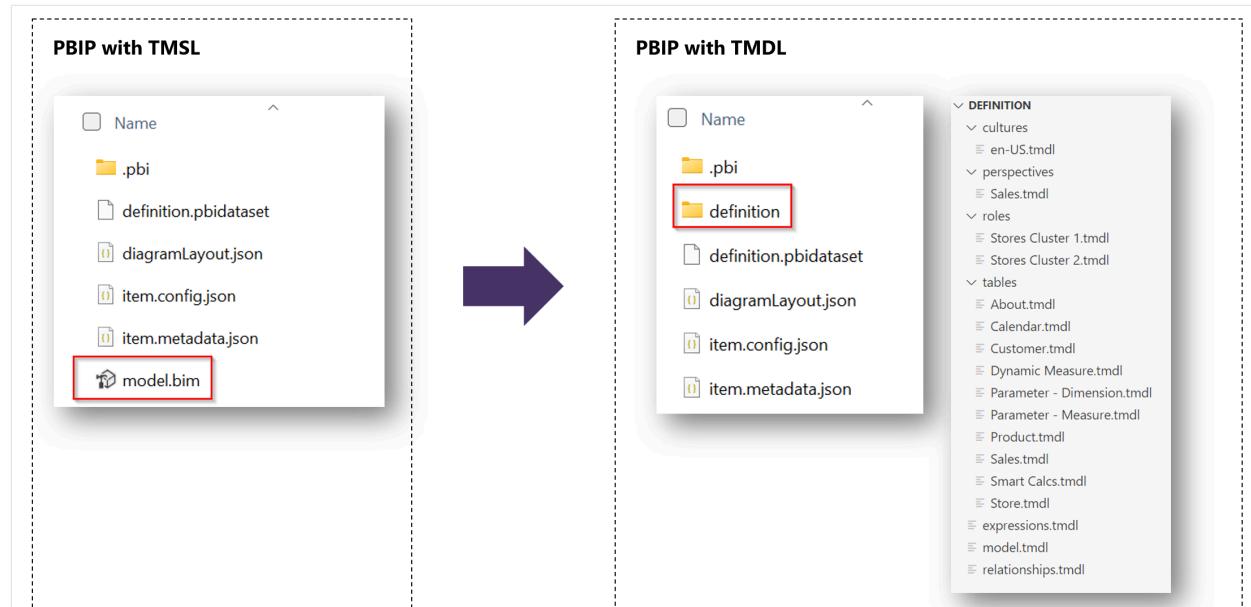
1. Open the PBIP in Power BI Desktop.
2. Ensure the Preview Feature is enabled.
3. **Save** the project. A prompt appears asking you to upgrade into TMDL.
4. Select **Upgrade**.

i **Important**

Once you upgrade to TMDL, you can't revert back to TMSL. If you think you might want to revert back to TMSL, save a copy of your PBIP files first.



The existing Tabular Model Scripting Language (TMSL) file (*model.bim*) is replaced with a `\definition` folder containing the TMDL representation of the semantic model.



If you select to **Keep current format**, Desktop won't prompt again to upgrade.

## Make external changes to TMDL files

For a better experience reading and editing your TMDL files, install the [TMDL - Visual Studio Marketplace](#) Microsoft VS Code extension.

Open the PBIP folder using VS Code and navigate to semantic model definition folder.

The screenshot shows the Power BI Desktop interface. On the left is the Explorer pane, which displays a tree view of project files under 'DEVMODE\_NEWFORMATS'. A red box highlights the 'tables' node, which contains several TMDL files: About.tmdl, Calendar.tmdl, Customer.tmdl, Dynamic Measure.tmdl, Parameter - Dimension.tmdl, Parameter - Measure.tmdl, Product.tmdl, Sales.tmdl, Smart Calcs.tmdl, database.tmdl, expressions.tmdl, model.tmdl, and relationships.tmdl. On the right is the main workspace, showing the content of the 'Product.tmdl' file. The code is a JSON-based schema for a table named 'Product'.

```
1 /////////////////////////////////////////////////////////////////// Product.Catalog
2 table Product
3   lineageTag: e9374b9a-faee-4f9e-b2e7-d9aafb9d6a91
4
5   measure '#.Products' = COUNTROWS('Product')
6   formatString: "#,##0"
7   lineageTag: 1f8f1a2a-06b6-4989-8af7-212719cf3617
8
9   column Product
10  dataType: string
11  lineageTag: da435585-1f9a-44bd-ba2c-34c98f298cfc
12  isDefaultLabel
13  summarizeBy: none
14  sourceColumn: Product
15
16  annotation SummarizationSetBy = Automatic
17
18  column ProductKey
19  dataType: int64
20  isKey
21  formatString: 0
22  isAvailableInMdx: false
23  lineageTag: 4184d53e-cd2d-4cbe-b8cb-04c72a750bc4
24  summarizeBy: none
25  sourceColumn: ProductKey
26
27  annotation SummarizationSetBy = Automatic
28
29  column 'Product.Code'
30  dataType: string
31  lineageTag: e9d204ad-76d8-4db9-9d1a-b9c07a4b50b2
32  summarizeBy: none
33  sourceColumn: Product.Code
34
35  annotation SummarizationSetBy = Automatic
36
```

Power BI Desktop isn't aware of changes to project files made by other tools. Therefore, if you make any changes to open files outside of Power BI Desktop, you need to restart for those changes to be shown in Power BI Desktop.

Please refer [here](#) for supported write operations outside of Power BI Desktop.

## TMDL Errors

If any invalid edits are made to the TMDL files, Power BI Desktop throws an error on open, with the location of the error:



## Unable to open file

There's a problem with the definition content in your Power BI Project.

TMDL Format Error:

Parsing error type - UnknownKeyword

Detailed error - Unsupported property - formaString is not a supported property in the current context!

Document - './tables/Product'

Line Number - 6

Line - ' formaString: #,##0'

[Copy details to clipboard](#)

[Close](#)

## TMDL considerations and limitations

During the Public Preview, [Fabric Git Integration](#) will still export the semantic model using TMSL by default. However, if the semantic model is imported into Fabric using Fabric Git Integration with TMDL format, then Fabric Git Integration will use TMDL format to export the semantic model definition to Git if there are any semantic model changes in the service.

### Important

If you import your semantic model using any other import method, such as [Power BI Desktop Publish](#), Fabric Git Integration will switch back to the default TMSL format.

## Related content

- [Power BI Desktop project report folder](#)
- [Power BI Desktop projects](#)
- [Tabular Model Scripting Language \(TMSL\)](#)

# Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Ask the community ↗](#)

# Power BI Desktop project report folder

Article • 08/13/2024

## ⓘ Important

Power BI Desktop projects is currently in preview.

This article describes the files and subfolders in a Microsoft Power BI Desktop project's **Report** folder. The files and subfolders here represent a Power BI report. Depending on your project, the report folder can include:

- .pbil\
  - localSettings.json
- CustomVisuals\
- StaticResources\
  - RegisteredResources\
- semanticModelDiagramLayout.json
- definition.pbir<sup>1</sup>
- mobileState.json
- report.json<sup>2</sup>
- definition\ folder<sup>3</sup>
- .platform

<sup>1</sup> - This file is required.

<sup>2</sup> - This file is required when saving to PBIR-Legacy format.

<sup>3</sup> - This file is required when saving to PBIR format.

Not every project report folder includes all of the files and subfolders described here.

## Report files

### .pbil\localSettings.json

Contains report settings that apply only for the current user and local computer. It should be included in gitignore or other source control exclusions. By default, Git ignores this file.

For more information, see the [localSettings.json schema document](#).

## **CustomVisuals\**

A subfolder that contains metadata for custom visuals in the report. Power BI supports three kinds of custom visuals:

- Organizational store visuals - Organizations can approve and deploy custom visuals to Power BI for their organization. To learn more, see [Organization store](#).
- AppSource Power BI visuals - Also known as "Public custom visuals". These visuals are available from Microsoft AppSource. Report developers can install these visuals directly from Power BI Desktop.
- Custom visual files - Also known as "Private custom visuals". The files can be loaded into the report by uploading a pbviz package.

Only private custom visuals are loaded into the CustomVisuals folder. AppSource and Organization visuals are loaded automatically by Power BI Desktop.

## **RegisteredResources\**

A subfolder that includes resource files specific to the report and loaded by the user, like custom themes, images, and custom visuals (pbviz files).

Developers are responsible for the files here and changes are supported. For example, you can change a file and after a Power BI Desktop restart, the new file is loaded into the report. This folder can unblock some useful scenarios, like:

- Authoring custom themes outside of Power BI Desktop by using the public schema.
- Applying batch changes by changing the resource file on multiple reports. For example, you can switch the corporate custom theme, change between light and dark themes, and change logo images.

Every resource file must have a corresponding entry in the report.json file, which during **preview** doesn't support editing. Edits to RegisteredResources files are only supported for already loaded resources that cause Power BI Desktop to register the resource in report.json.

## **semanticModelDiagramLayout.json**

Contains data model diagrams describing the structure of the semantic model associated with the report. During **preview**, this file doesn't support external editing.

## **definition.pbir**

Contains the overall definition of a report and core settings. This file also holds the reference to the semantic model used by the report. Power BI Desktop can open a pbir file directly, just the same as if the report were opened from a pbip file. Opening a pbir also opens the semantic model alongside if there's a relative reference using `byPath`.

Example definition.pbir:

```
JSON

{
  "version": "1.0",
  "datasetReference": {
    "byPath": {
      "path": "../Sales.Dataset"
    },
    "byConnection": null
  }
}
```

The definition includes the `datasetReference` property, which references the semantic model used in the report. The reference can be either:

`byPath` - Specifies a relative path to the target semantic model folder. Absolute paths aren't supported. A forward slash (/) is used as a folder separator. When used, Power BI Desktop also opens the semantic model in full edit mode.

`byConnection` - Specifies a remote semantic model in the Power BI service by using a connection string. When a `byConnection` reference is used, Power BI Desktop doesn't open the semantic model in edit mode.

Using a `byConnection` reference, the following properties must be specified:

[+] Expand table

Property	Description
connectionString	The connection string referring to the remote semantic model.
pbiModelDatabaseName	The remote semantic model ID.
connectionType	Type of connection. For service remote semantic model, this value should be <code>pbiServiceXmlaStyleLive</code> .
pbiModelVirtualServerName	An internal property that should have the value, <code>sobe_wowvirtualserver</code> .

Example using `byConnection`:

```
JSON

{
    "version": "1.0",
    "datasetReference": {
        "byPath": null,
        "byConnection": {
            "connectionString": "Data
Source=powerbi://api.powerbi.com/v1.0/myorg/WorkspaceName;Initial
Catalog=SemanticModelName;Integrated Security=ClaimsToken",
            "pbiServiceModelId": null,
            "pbiModelVirtualServerName": "sobe_wowvirtualserver",
            "pbiModelDatabaseName": "e244efd3-e253-4390-be28-6be45d9da47e",
            "connectionType": "pbiServiceXmlaStyleLive",
            "name": null
        }
    }
}
```

When the semantic model and report share the same workspace, [Fabric Git Integration](#) always uses a `byPath` reference to the semantic model.

This file also specifies the supported report definition formats through the 'version' property.

[ ] [Expand table](#)

Version	Supported formats
1.0	Report definition must be stored as PBIR-Legacy in the report.json file.
4.0 or above	Report definition can be stored as PBIR-Legacy (report.json file) or <a href="#">PBIR</a> (\definition folder).

For more information, see the [definition.pbir schema document](#).

## mobileState.json

Contains report appearance and behavior settings when rendering on a mobile device. This file doesn't support external editing.

## report.json

This file contains the report definition in the Power BI Report Legacy format (PBIR-Legacy) and doesn't support external editing.

## definition\ folder

This folder is only available if the Power BI project is saved using the [Power BI enhanced report format \(PBIR\)](#). It replaces the `report.json` file.

### .platform

Fabric platform file that holds properties vital for establishing and maintaining the connection between Fabric items and Git.

To learn more, see [Git integration automatically generated system files](#).

## PBIR format

### Important

Please consider all the PBIR [limitations](#) during the preview phase.

Saving your Power BI Project files (PBIP) using the Power BI Enhanced Report Format (PBIR) greatly improves change tracking and merge conflict resolution by using properly formatted JSON files.

```
1  {
2    "$schema": "https://developer.microsoft.com/json-schemas/fabric/item/report/definition/
visualContainer/1.0.0/schema.json",
3    "name": "7c4826a02220bab75601",
4    "position": {
5      "x": 85.37883561975859,
6      "y": 33.754423384555722,
7      "z": 0,
8      "width": 299.81870182752436,
9      "height": 299.81870182752436
10    },
11    "visual": {
12      "visualType": "card",
12+     "visualType": "gauge",
13      "query": {
14        "queryState": {
15          "Values": {
15+            "Y": {
```

Each page, visual, bookmark, etc., is organized into a separate, individual file within a folder structure. This format is ideal for co-development conflict resolution.

```
✓ Finance.Report
  > .pbix
  ✓ definition
    ✓ bookmarks
      {} bookmarks.json
      {} c6213a156e9d09533069.bookmark.json
    ✓ pages
      ✓ b516d33b77c4b87e39cd
        ✓ visuals\91521e2041a027b3d008
          {} visual.json
          {} page.json
        ✓ c4e1d722a96b80877b3c
          > visuals
          {} page.json
          {} pages.json
        {} report.json
        {} version.json
      > StaticResources
    ≡ .platform
    ≡ definition.pbir
```

Unlike PBIR-Legacy (report.json), PBIR is a publicly documented format that supports modifications from non-Power BI applications. Each file has a public JSON schema, which not only documents the file but also lets code editors like Visual Studio Code perform syntax validation while editing.

Some of the possible scenarios now available with PBIR include:

- Copy pages/visuals/bookmarks between reports.
- Ensure consistency of a set of visuals across all pages, by copying & pasting the visual files.
- Easy find and replace across multiple reports files.
- Apply a batch edit across all visuals using a script (for example, hide visual level filters)

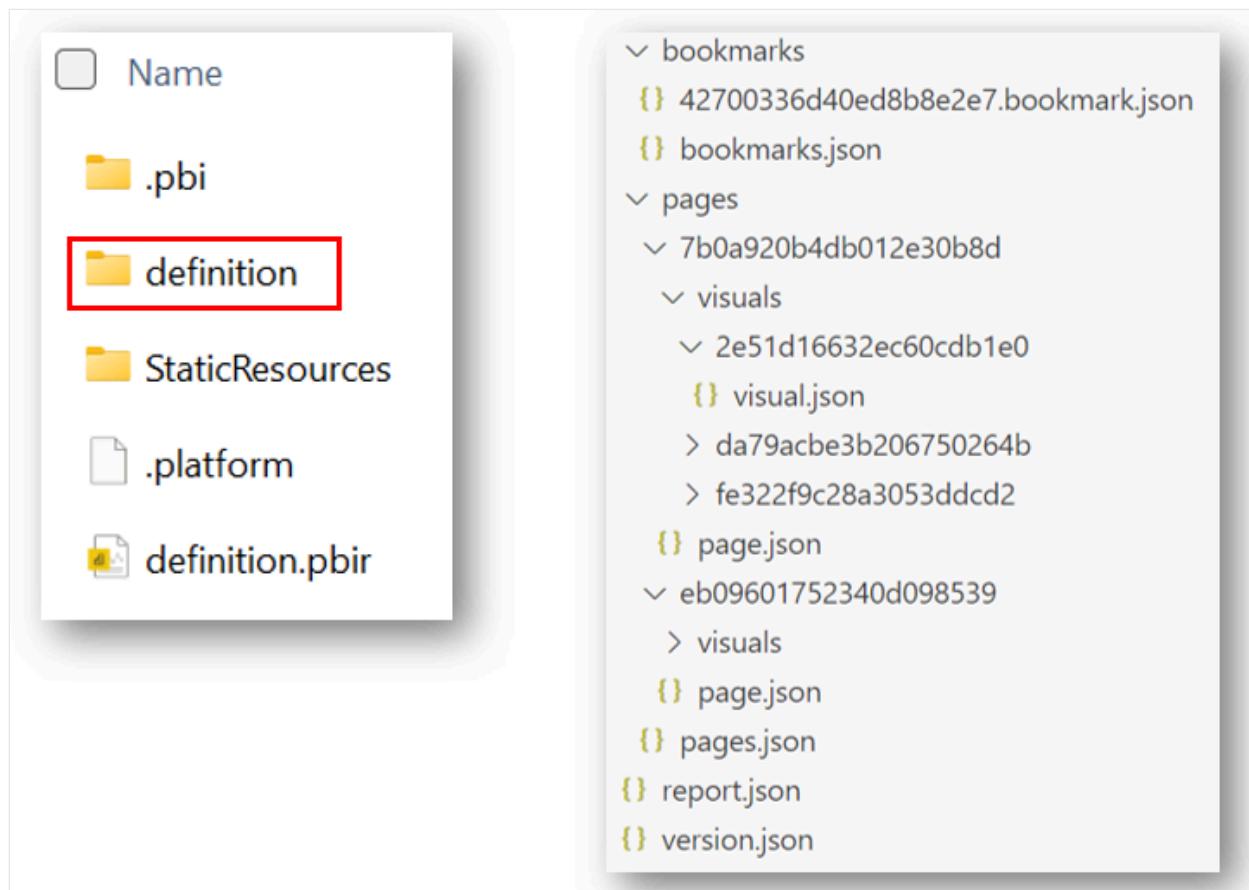
## Enable PBIR format Preview feature

Saving as a Power BI Project using PBIR is currently in preview. Before using it, enable it in Power BI Desktop preview features:

Go to **File > Options and settings > Options > Preview features** and check the box next to **Store reports using enhanced metadata format (PBIR)**.

## Save as a project using PBIR

With the PBIR Preview feature enabled, when you save a project, your report is saved within a folder named `\definition` inside of [report folder](#):



Learn more about the [PBIR folder structure](#).

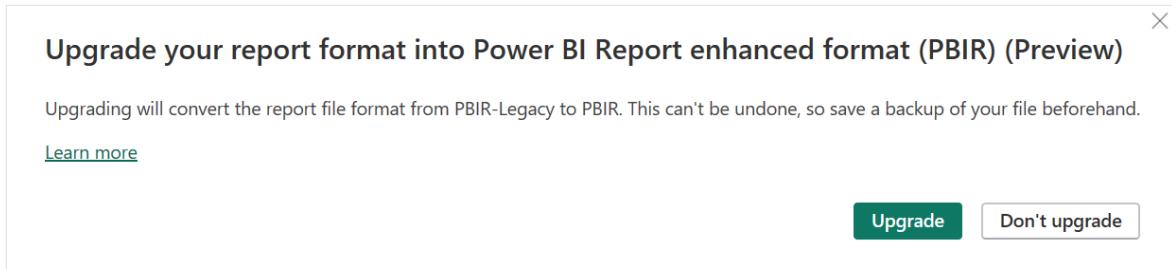
## Convert existing PBIP to PBIR

If you already have a PBIP using PBIR-Legacy format, you can convert it to PBIR as follows:

1. Open the PBIP in Power BI Desktop.
2. Ensure the Preview Feature is enabled.

3. Save the project. A prompt appears asking you to upgrade into PBIR.

4. Select **Upgrade**.



### ⓘ Important

Once you upgrade to PBIR, you can't revert back to PBIR-Legacy. If you think you might want to revert back to PBIR-Legacy, save a copy of your PBIP files first.

The existing PBIR-Legacy file (`report.json`) is replaced with a `\definition` folder containing the PBIR representation of the report.

If you select to **Keep current format**, Desktop don't prompt again to upgrade.

## Publish a PBIR report to service

While in the preview phase, the only way to publish a report with the PBIR format is through [Fabric Git Integration](#). This involves connecting the workspace to a Git repository and pushing the PBIR report to it, which can then be synchronized with the service workspace at a later stage.

If you wish to convert an existing report to PBIR in the service, follow these steps:

1. Connect your workspace to Git.
2. Clone the Git repository to your local file system.
3. Open the report in Power BI Desktop by opening the `definition.pbir` file.
4. Save the report and choose to upgrade to PBIR.
5. Commit and sync the changes to Git.
6. Update the workspace with the latest changes from Git.

## PBIR folder and files

The report definition is stored inside the `definition\` folder with the following structure:

```

└── bookmarks\
    ├── [bookmarkName].bookmark.json
    └── bookmarks.json
└── pages\
    ├── [pageName]\
        ├── \visuals
        │   ├── [visualName]\
        │   │   ├── mobile.json
        │   │   └── visual.json
        └── page.json
    └── pages.json
└── version.json
└── reportExtensions.json
└── report.json

```

[\[+\] Expand table](#)

File/Folder	Required	Description
bookmarks\	No	Folder holding all bookmark files of the report.
— [bookmarkName].bookmark.json	No	Bookmark metadata, such as target visuals and filters. More information at <a href="#">schema ↗</a> .
— bookmarks.json	No	Bookmarks metadata, such as bookmark order and groups. More information at <a href="#">schema ↗</a> .
pages\	Yes	Folder holding all pages of the report.
— [pageName]\	Yes	One folder per page.
—— visuals\	No	Folder holding all visuals of the page.
—— [visualName]\	No	One folder per visual.
——— mobile.json	No	Visual mobile layout metadata, such as mobile position and formatting. More information at <a href="#">schema ↗</a> .
——— visual.json	Yes	Visual metadata, such as position and formatting, query. More information at <a href="#">schema ↗</a> .
——— page.json	Yes	Page metadata, such as page level filters and formatting. More information at <a href="#">schema ↗</a> .
——— pages.json	No	Pages metadata, such as page order and active page.

File/Folder	Required	Description
More information at <a href="#">schema ↗</a> .		
version.json	Yes	PBIR file version, among other factors, determines the required files to be loaded. More information at <a href="#">schema ↗</a>
reportExtensions.json	No	Report extensions, such as report level measures. More information at <a href="#">schema ↗</a>
report.json	Yes	Report metadata, such as report level filters and formatting. More information at <a href="#">schema ↗</a>

## PBIR naming convention

All names inside the square brackets ([]) in the preceding table follow a default naming convention but can be renamed to more user-friendly names. By default, pages, visuals, and bookmarks use their report object name as their file or folder name. These object names are initially a 20-character unique identifier, such as '90c2e07d8e84e7d5c026'.



```

1   {
2     "$schema": "https://developer.microsoft.com/json-schemas/reporting/report.json#",
3     "name": "91521e2041a027b3d008",
4     "position": {
5       "x": 428.4581550850744,
6       "y": 247.11086334771437,
7       "width": 300,
8       "height": 300,
9       "z": 0
10    },
11    ...
  
```

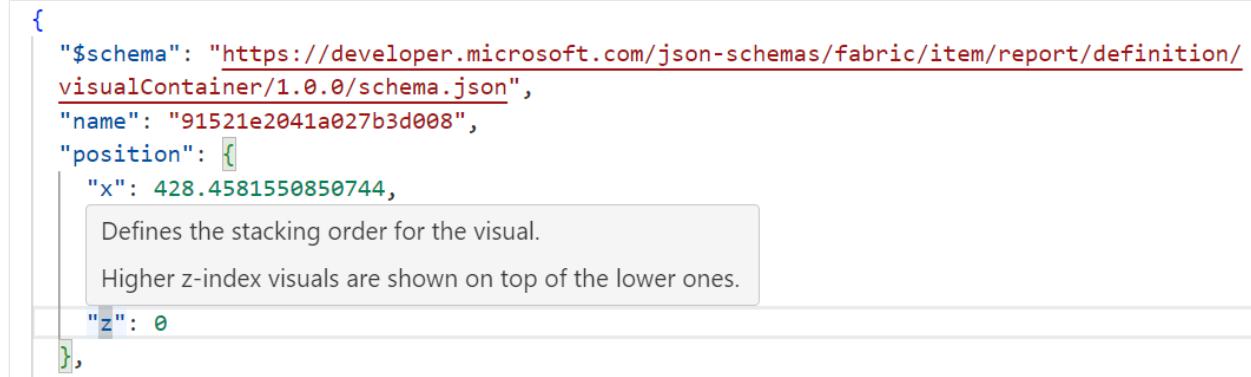
Renaming the 'name' property within each JSON file is supported but might break external references both inside and outside the report. The object name and/or file/folder name must consist of one or more word characters (letters, digits, underscores) or hyphens.

After renaming any PBIR files or folders, you must restart Power BI Desktop. Upon restart, Power BI Desktop will preserve the original file or folder names when saving.

## PBIR Json Schemas

Each PBIR JSON file includes a [JSON schema ↗](#) declaration at the top of the document. This schema URL is publicly accessible and can be used to learn more about the

available properties and objects for each file. Additionally, it provides built-in IntelliSense and validation when editing with code editors like [Visual Studio Code](#).



The screenshot shows a JSON object with several properties. The 'x' property is highlighted with a tooltip explaining it defines the stacking order for the visual, with higher z-index values appearing on top. The 'z' property is also shown.

```
{
  "$schema": "https://developer.microsoft.com/json-schemas/fabric/item/report/definition/visualContainer/1.0.0/schema.json",
  "name": "91521e2041a027b3d008",
  "position": [
    {
      "x": 428.4581550850744,
      "z": 0
    }
  ]
}
```

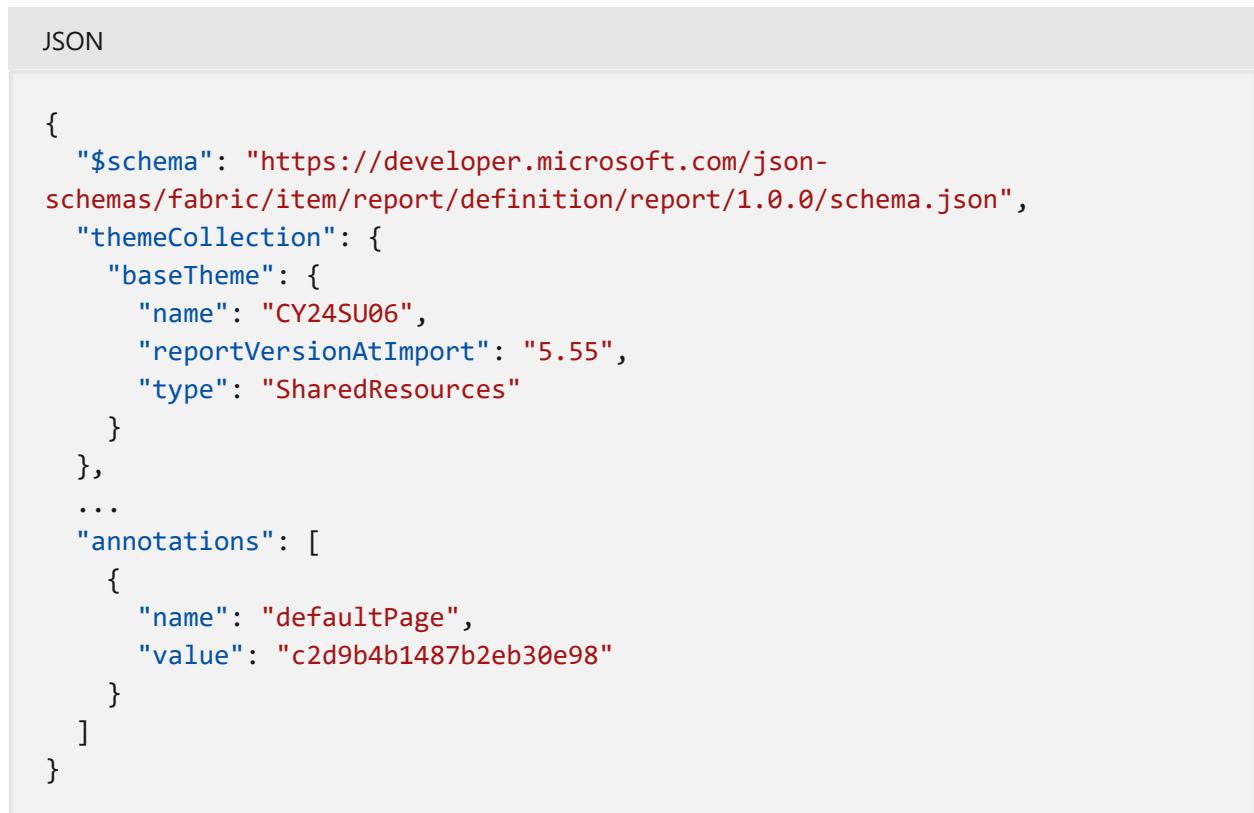
The schema URL also defines the version of the document, which is expected to change as the report definition evolves.

All the JSON schemas are published [here](#).

## PBIR annotations

You can include annotations as name-value pairs within the report definition for each `visual`, `page` and `report`. While Power BI Desktop will ignore these annotations, they can be valuable for external applications like scripts.

For instance, you could specify the `defaultPage` for the report at the `report.json` file, which can then be utilized by a deployment script.



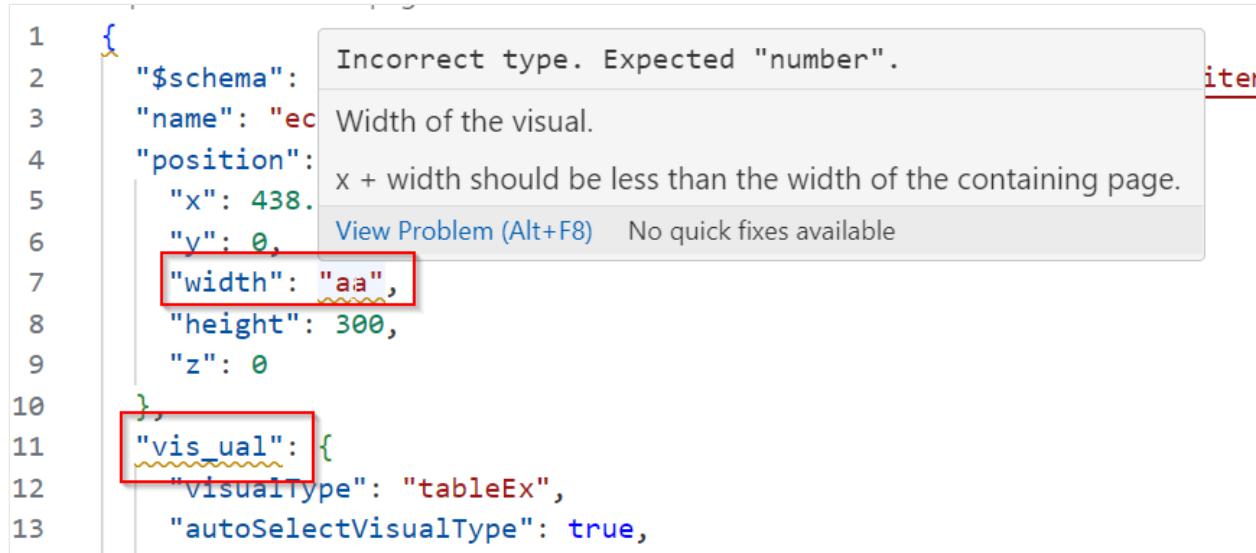
The screenshot shows a JSON object with an 'annotations' array containing one item. This item has a 'name' of 'defaultPage' and a 'value' of 'c2d9b4b1487b2eb30e98'.

```
JSON

{
  "$schema": "https://developer.microsoft.com/json-schemas/fabric/item/report/definition/report/1.0.0/schema.json",
  "themeCollection": {
    "baseTheme": {
      "name": "CY24SU06",
      "reportVersionAtImport": "5.55",
      "type": "SharedResources"
    }
  },
  ...
  "annotations": [
    {
      "name": "defaultPage",
      "value": "c2d9b4b1487b2eb30e98"
    }
  ]
}
```

## External changes to PBIR files

You can edit the PBIR JSON files using a code editor like [Visual Studio Code](#) or an external tool, as long as the file obey the JSON schema. Using a wrong property name or type can be easily detected directly in Visual Studio Code:



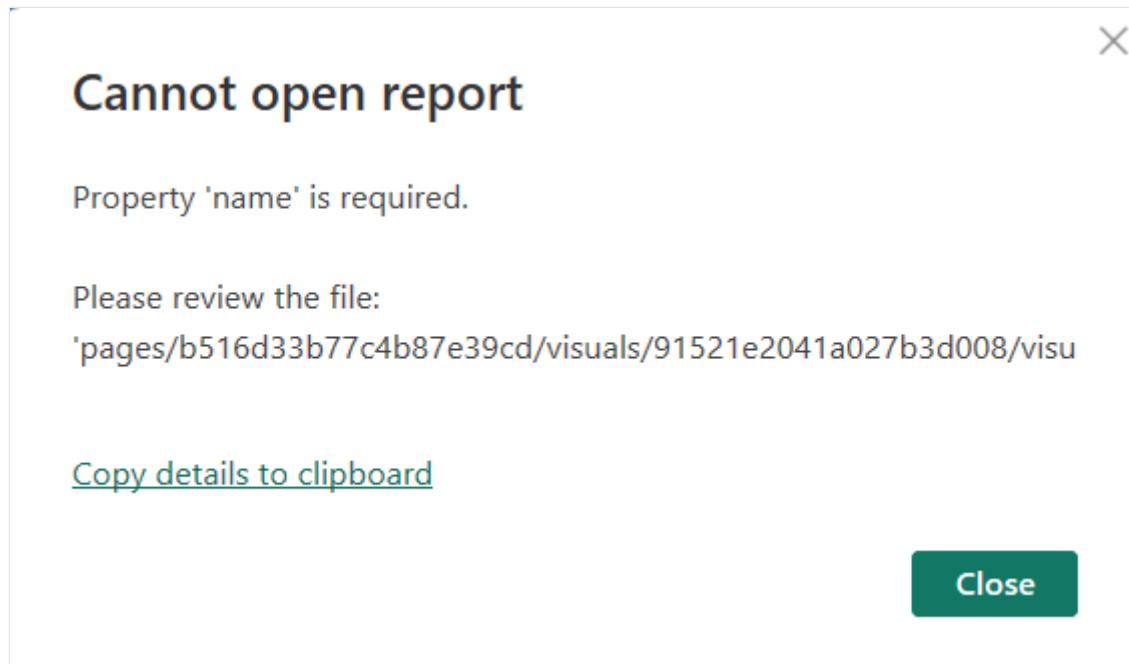
A screenshot of Visual Studio Code showing a JSON file with validation errors. The file content is:

```
1  {
2      "$schema": "http://powerbi.com/pbix-schema",
3      "name": "ec",
4      "position": {
5          "x": 438,
6          "y": 0,
7          "width": "aa", // Error: Incorrect type. Expected "number".
8          "height": 300,
9          "z": 0
10 },
11     "visual": { // Error: Incorrect type. Expected "string".
12         "visualtype": "tableEx",
13         "autoSelectVisualType": true,
```

The line "width": "aa" has a red box around it, and a tooltip says "Incorrect type. Expected "number".". The line "visual": { has a red box around it, and a tooltip says "Incorrect type. Expected "string".".

External changes to PBIR content might result in errors when reopening the files in Power BI Desktop. These errors can be of two types:

**Blocking errors** prevent Power BI Desktop from opening the report. These errors help identify the issue and the offending file that must be fixed before reopening:



Errors such as an invalid schema or missing required properties are considered blocking errors. Those errors can be easily identified by opening the file in Visual Studio Code and inspecting the schema errors.

Non-blocking errors don't prevent Power BI Desktop from opening the report and are automatically resolved.

**Your report had issues that have been resolved.** ×

We encountered some issues in the report definition and resolved them. We can continue and open the report, but if you save, there may be unexpected changes

**List of applied fixes (1):**

- ActivePageName not found

[Copy details to clipboard](#)

Continue Close Desktop

Errors such as an invalid `activePageName` configuration are examples of nonblocking errors that are automatically fixed. The warning is necessary to give you the chance to avoid saving the report with the autofix, by that preventing any potential loss of work.

## Common PBIR errors

**Scenario:** *After rename visual or page folder names, my visual or page no longer appears when opening the report.*

**Solution:** Verify whether the name complies with the [naming convention](#). If it doesn't, Power BI Desktop ignores the file, or folder and treats it as private user files.

**Scenario:** *New report objects are named differently from others. For example, most page folders are named 'ReportSection0e71dafbc949c0853608', while a few are named '1b3c2ab12b603618070b'.*

**Solution:** PBIR adopted a new [naming convention](#) for every object, but it only applies to new objects. When you save an existing report as PBIP, the current names must be preserved to prevent breaking references. If you want consistency, a script a batch rename is allowed.

**Scenario:** *I copied a bookmark file, and upon saving, most of the bookmark configuration was deleted.*

**Solution:** This behavior is intentional, report bookmarks capture the state of a report page along with all its visuals. Since the captured state originates from another report page with different visuals, any invalid visuals are removed from the bookmark

configuration. If you also copy the dependent visuals and page, the bookmark maintains its configuration.

**Scenario:** I copied a page folder from another report and encountered an error stating, "Values for the 'pageBinding.name' property must be unique."

**Solution:** The pageBinding object is necessary to support drillthrough and page tooltips. Since they might be referenced by other pages, the name must be unique within the report. On the newly copied page, assign a unique value to resolve the error. After June 2024, this situation is no longer an issue because the pageBinding name is a GUID by default.

## PBIR considerations and limitations

PBIR is currently in preview. Keep the following in mind:

- Service limitations
  - Can't be included in Power BI Apps.
  - Can't be downloaded as PBIX.
  - Can't be deployed with deployment pipelines.
  - Can't be saved as a copy.
  - Can't be published from Power BI Desktop.
  - Can't be uploaded to workspace as PBIX.
- Large reports with more than 500 files experience authoring performance issues (report viewing isn't affected), including:
  - Saving in Power BI Desktop
  - Synchronization in Fabric Git Integration
- Once a report is converted from PBIR-Legacy to PBIR, it isn't possible to roll it back.
- Converting a PBIP file to a PBIX file using the "Save As" feature embeds the PBIR report within the PBIX file, carrying over all PBIR limitations to the PBIX.

PBIR size limitations enforced by the service:

- 1,000 max pages per report.
- 300 max visuals per page.
- 5 mb max for each bookmark file.
- 1 mb max for each file.
- 1,000 max resource package files per report.
- 300 mb max size for all resource package files.
- 20 mb max size of all report files.

During the Public Preview, [Fabric Git Integration](#) and [Fabric REST APIs](#) continue to use PBIR-Legacy (report.json) when exporting the report definitions. However, if the report is

imported into Fabric using PBIR format, then both features start exporting the report definition using PBIR format.

## Related content

- [Power BI Desktop project semantic model folder](#)
  - [Power BI Desktop projects](#)
- 

## Feedback

Was this page helpful?



Yes



No

[Provide product feedback](#) | [Ask the community](#)

# Power BI Desktop projects Git integration

Article • 08/13/2024

## ⓘ Important

Power BI Desktop projects is currently in [preview](#).

Git integration in Microsoft Visual Studio Code (VS Code) enables Pro BI developers working with Power BI Desktop projects to streamline development processes, source control, and collaboration with Git repositories.

With Git integration, you can:

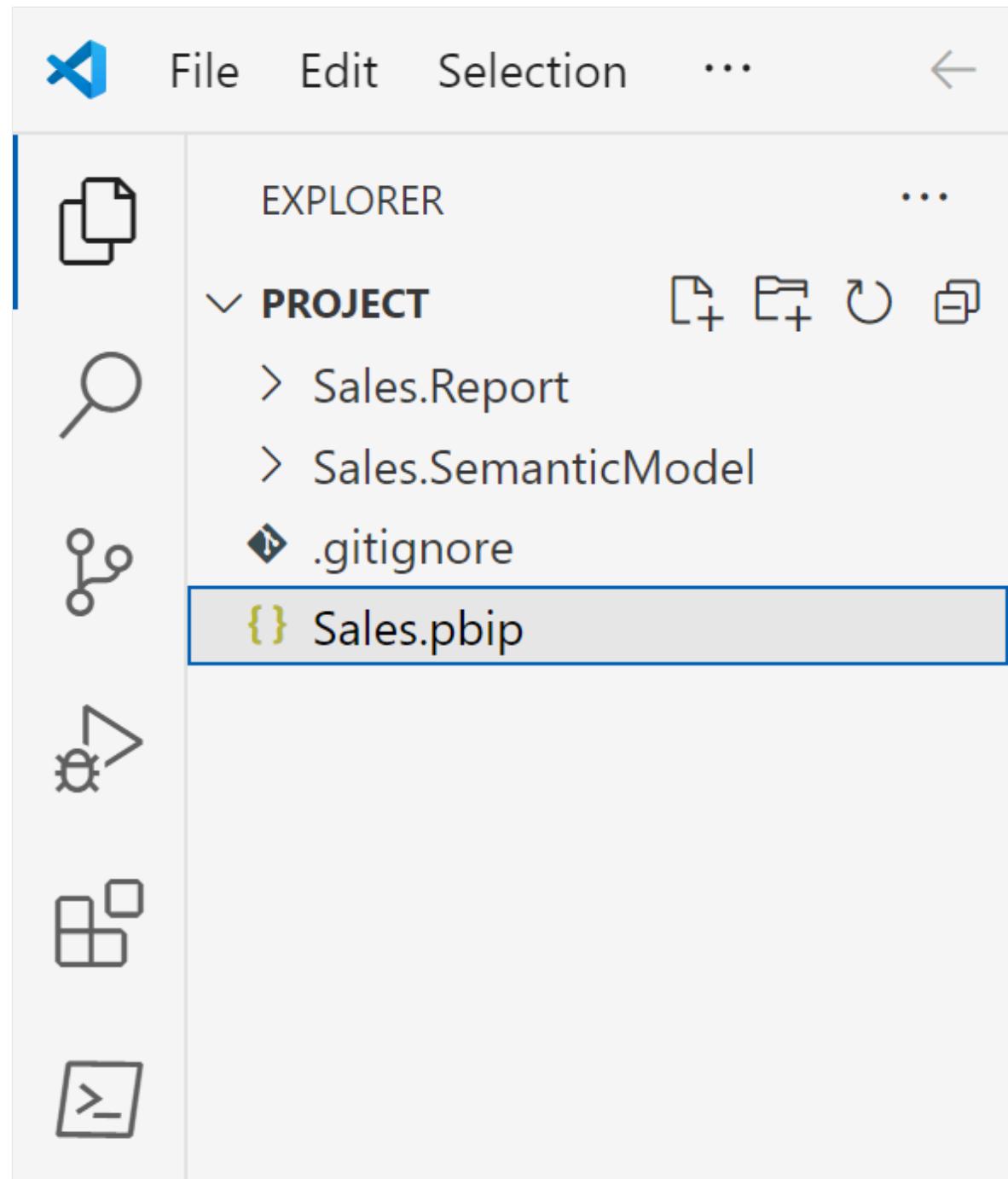
- Backup and version your work.
- Revert to previous states.
- Collaborate with others or work alone using Git branches.
- Use the capabilities of familiar source control tools, like Azure DevOps.

## Prerequisites

- Familiarity with Git. See [Git and GitHub learning resources](#).
- [Download](#) and install Git.
- [Download](#) and install VS Code development environment. It has native integration with Git. To learn more, see [Using Git source control in VS Code](#).

## Create a local Git repo using VS Code

1. In VS Code, open a Power BI Desktop project folder:



2. Initialize a Git repository by selecting **Source Control > Initialize Repository**:



File Edit Selection ...



## SOURCE CONTROL



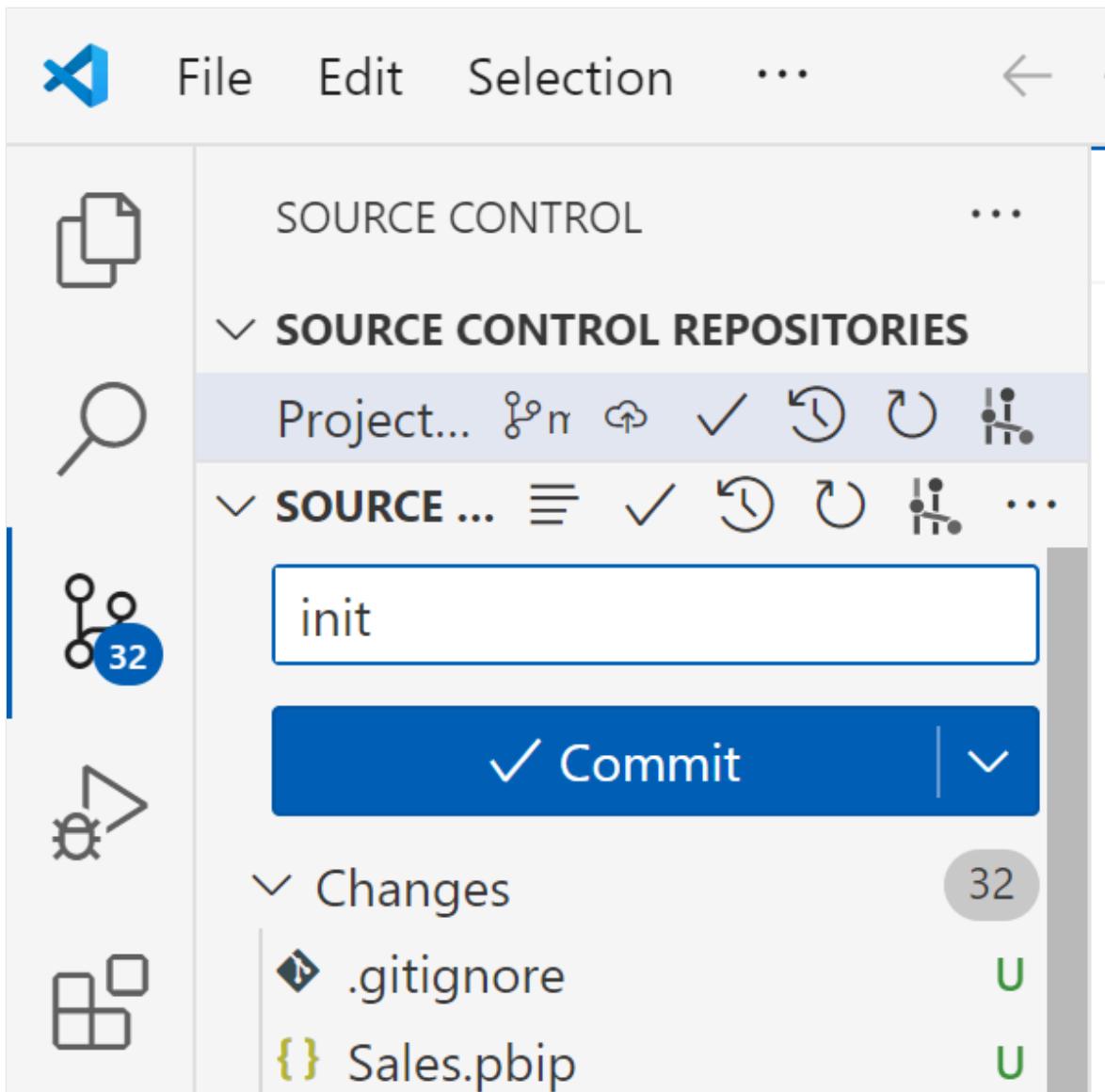
The folder currently open  
doesn't have a Git repository.  
You can initialize a repository  
which will enable source control  
features powered by Git.



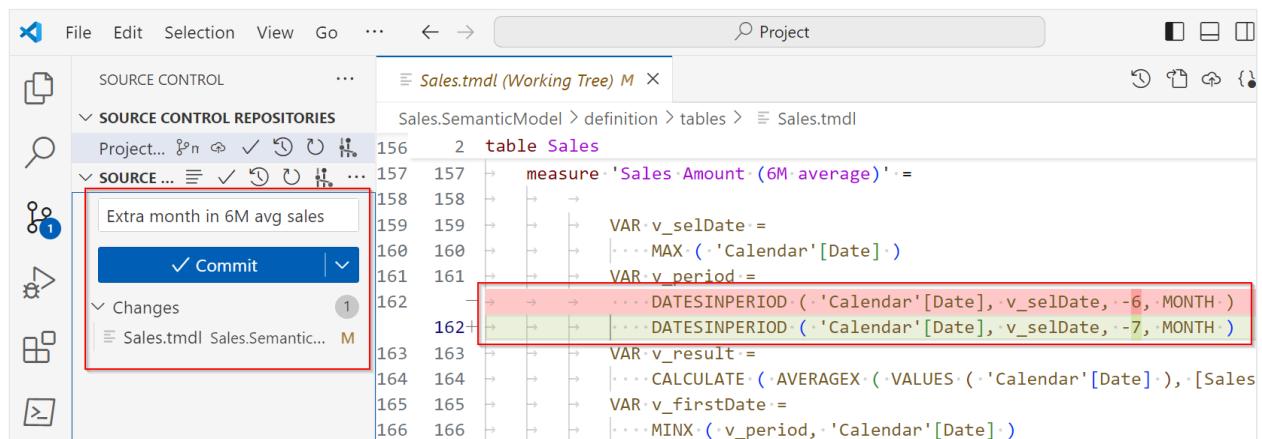
Initialize Repository

To learn more about how to use  
Git and source control in VS  
[Code read our docs.](#)

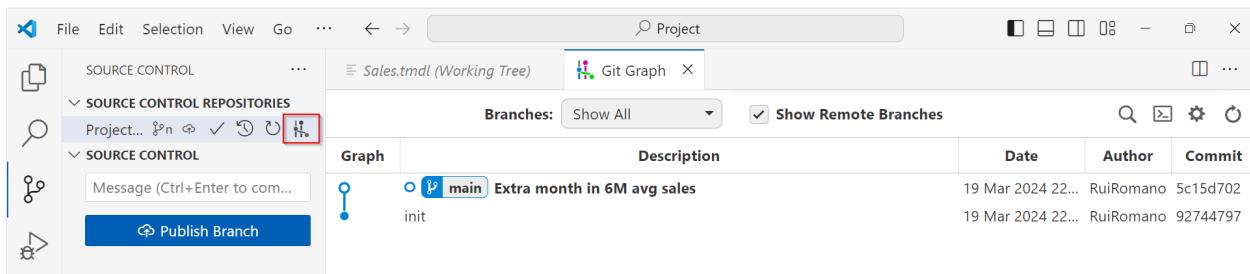
3. Do an initial Commit and enter a message:



From now on, any changes you make in Power BI Desktop changes a file in the folder tracked by your local Git. For example, in Power BI Desktop, when you change a DAX formula for a measure and then save, it triggers a Git diff on the model.bim file.



With Git integration, you can not only backup your work, but also track your change history. For example, with GitGraph, a popular free VS Code extension, you can easily track all your changes.



## Related content

- [Power BI Desktop projects Azure DevOps integration](#)
  - [Power BI Desktop project semantic model folder](#)
  - [Power BI Desktop project report folder](#)
- 

## Feedback

Was this page helpful?

[Yes](#)

[No](#)

[Provide product feedback ↗](#) | [Ask the community ↗](#)

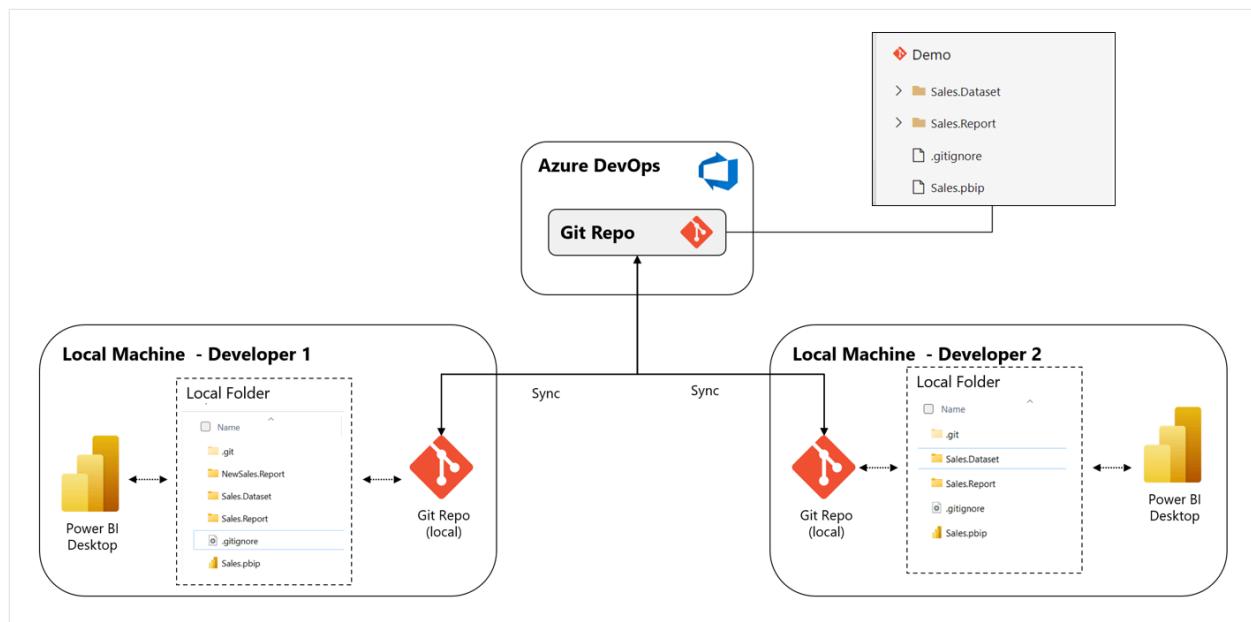
# Power BI Desktop projects Azure DevOps integration

Article • 08/19/2024

## ⓘ Note

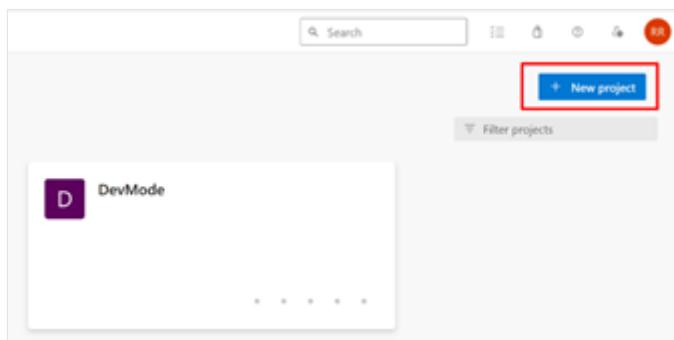
Power BI Desktop projects is currently in preview.

Microsoft Power BI Desktop projects enable developer collaboration by connecting your local Git repo to a remote Git host, like Azure DevOps.

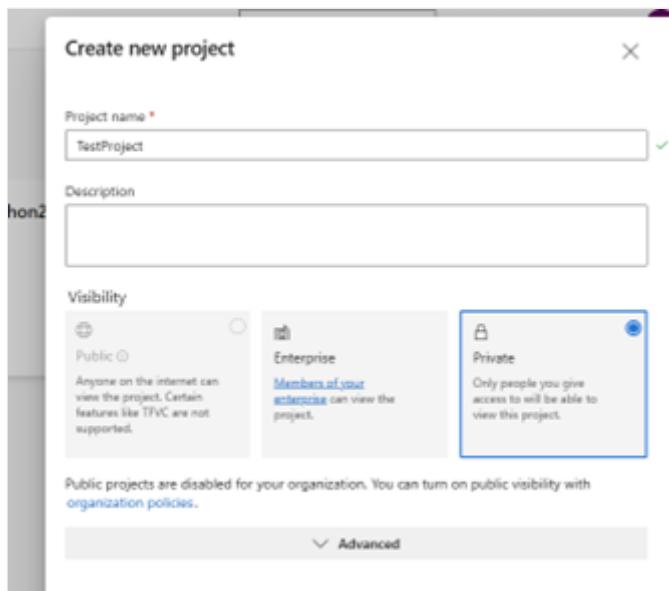


## Create a Git repo in Azure DevOps

1. In [Azure DevOps](#), select an existing organization, or create a new one.
2. Create a new Project within the organization:



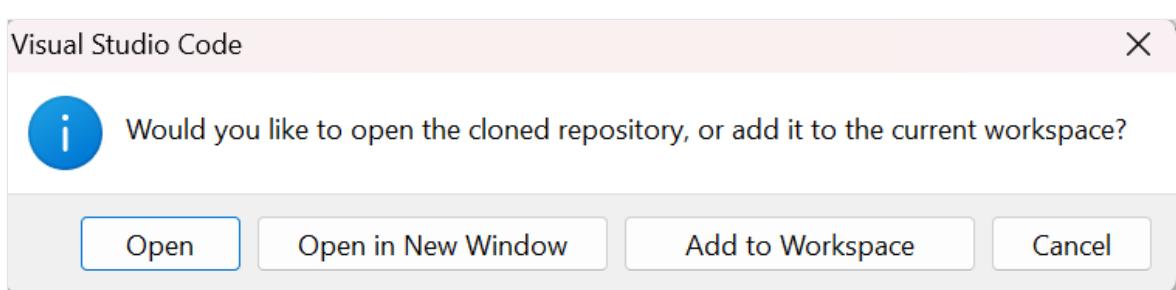
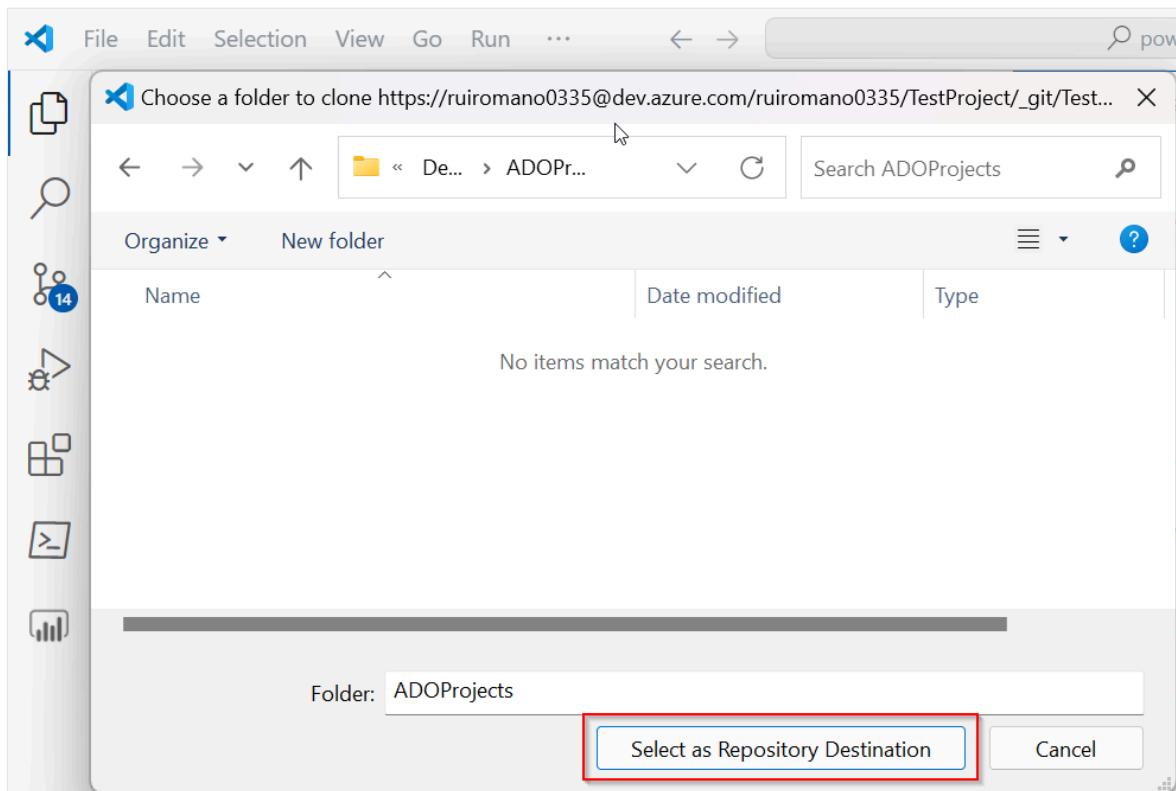
3. Enter your project details.



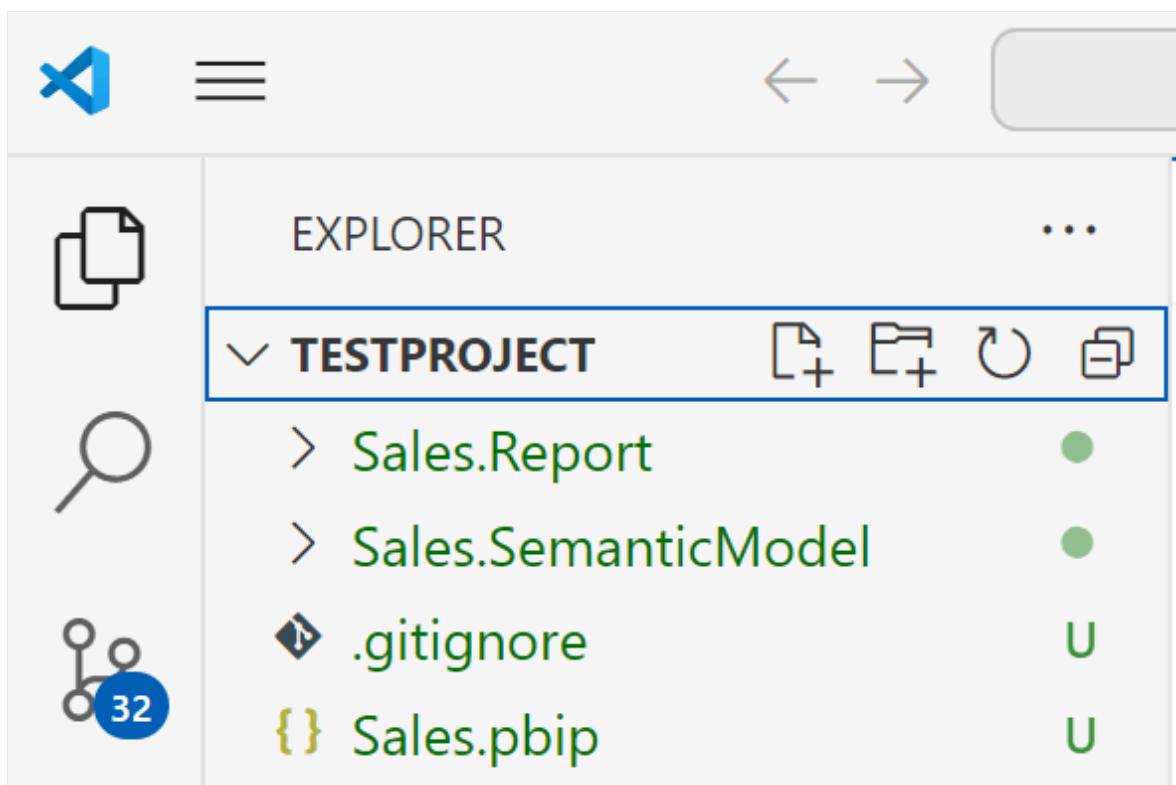
4. Select **Repos > Files**, and then select 'Initialize' to create an empty branch:

5. In **Repos > Files**, select **Clone > Clone in VS Code**

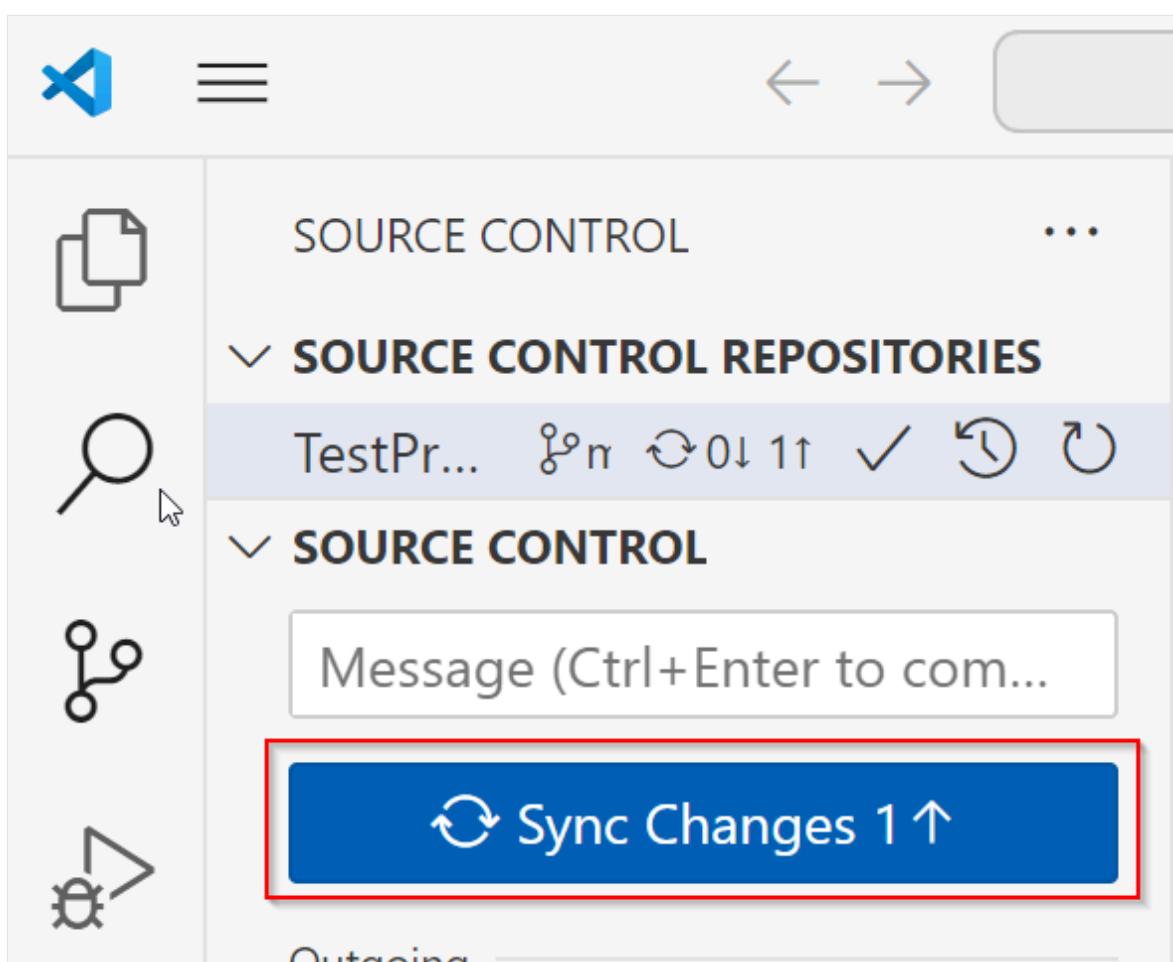
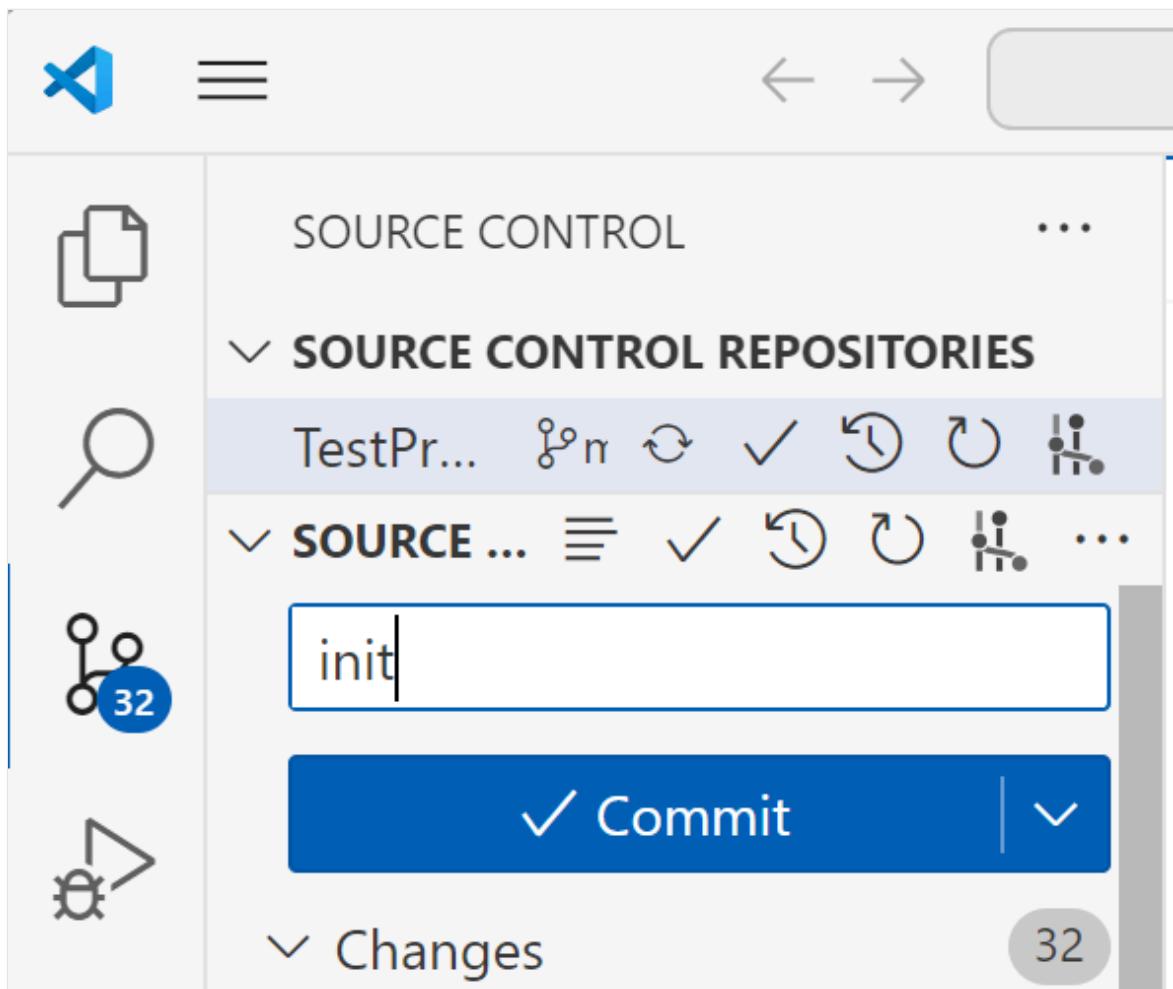
6. In **Visual Studio Code (VS Code) > pick a folder and open**



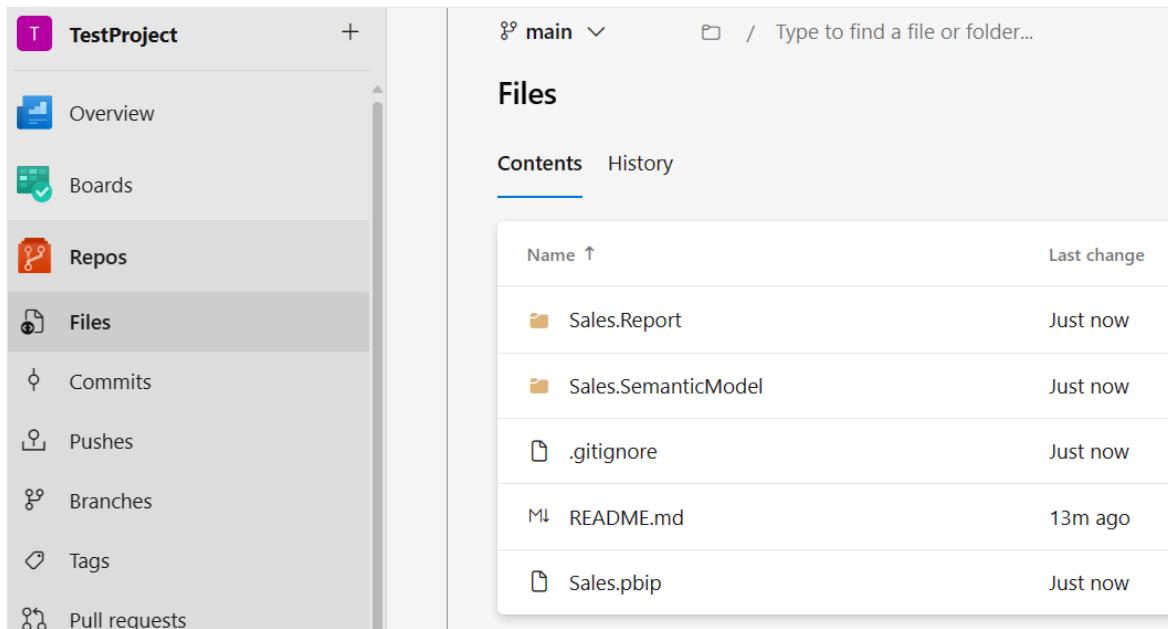
7. Save your Power BI Project files to the selected folder



## 8. Commit and sync changes



VS Code takes care of publishing your project into Azure DevOps, where you can see your project files.



As you can see, with Azure DevOps integration, you can have multiple developers working on the same Power BI project as long as they're synced with the same Azure Devops Git Repo.

If you're using Microsoft Fabric, you can also connect a Fabric workspace to an Azure DevOps Git repo and get all your content automatically deployed into the service. Git and Azure DevOps integration can provide a continuous integration workflow not only from Power BI Desktop to the service, but also from changes made in the service to Power BI Desktop. To learn more, see [Microsoft Fabric - Introduction to git integration](#).

## Related content

- [Power BI Desktop projects Git integration](#)
- [Power BI Desktop projects](#)

## Feedback

Was this page helpful?

Yes

No

[Provide product feedback ↗](#) | [Ask the community ↗](#)

# Power BI Project (PBIP) and Azure DevOps build pipelines for validation

Article • 12/09/2024

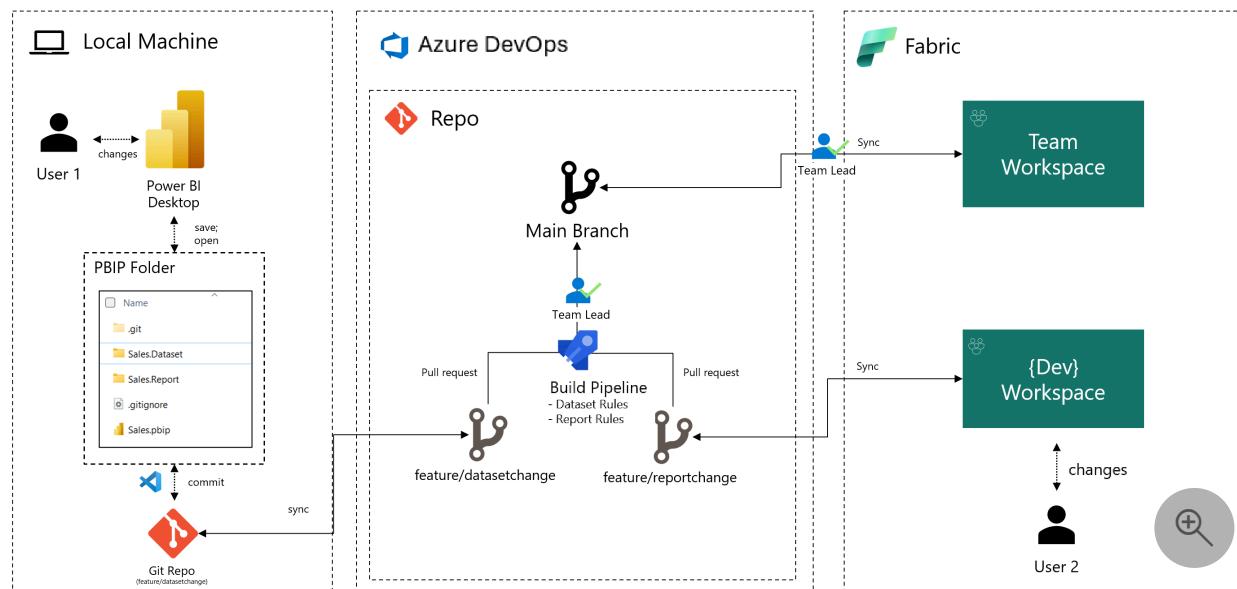
Combining Fabric Git Integration with Azure DevOps, enables you to connect a workspace to a branch in an Azure DevOps repository and automatically synchronizes between them.

Integrating the PBIP format with Azure DevOps lets you use Azure Pipelines to automate [Continuous Integration/Continuous Deployment](#) (CI/CD) pipelines. These pipelines process the PBIP metadata files and apply a series of quality checks to your development before deploying it to the production system.

In this article, we focus on continuous integration and describe how to create an Azure DevOps pipeline that guarantees best practices for all semantic models and reports within a Fabric workspace. By implementing automated quality tests, you can prevent common mistakes, and enhances team efficiency. For example, this approach ensures that new team members adhere to established standards for semantic model and report development.

Learn more about PBIP and Fabric Git Integration in [project-overview](#) and [Fabric Git integration overview](#).

The following diagram illustrates the end-to-end scenario with two development workflows that trigger the Azure DevOps pipeline to validate development quality. The pipeline execute does the following actions:



1. User 1 develops using Power BI Desktop.

- a. Create a branch from main using **VS Code** (feature/datasetchange)
  - b. Make changes to semantic model using Power BI Desktop
  - c. Commit changes to remote repository branch using **VS Code**
  - d. Create Pull Request to main branch using Azure DevOps
2. At the same time, *User 2* develops [using another Fabric workspace](#).
    - a. Create branch from main using Fabric Git (feature/reportchange)
    - b. Make report changes in the Fabric workspace
    - c. Commit changes to remote repository branch using Fabric Git
    - d. Create Pull Request to main branch using Azure DevOps
  3. The team lead reviews the Pull Requests and synchronizes the changes to the team workspace using Fabric Git.
  4. The Pull Request triggers the Azure DevOps pipeline to inspect the semantic model and report development quality.

#### Note

In this example, the pipeline uses two open-source community tools that enable a developer to apply (customizable) best practice rules to the metadata of semantic models and reports within a Power BI Project folder:

- [Tabular Editor](#) ↗ and [Best Practice Rules](#) ↗
- [PBI Inspector](#) ↗

An approach similar to the example in this article would apply to other community tools. This article doesn't delve into the specifics of the community tools, nor rule creation and editing. For in-depth information on these topics, refer to the links provided. The focus of this article is on the *process* of establishing a quality gate between source control and a Fabric Workspace. It's important to note that the referred community tools are developed by third-party contributors, and Microsoft doesn't offer support or documentation for them.

## Step 1 - Connect Fabric workspace to Azure DevOps

[Connect your Fabric workspace to Azure DevOps](#):

The screenshot shows the 'Workspace settings' page in Power BI. On the left, there's a sidebar with various icons for Home, Create, Browse, OneLake data hub, Apps, Metrics, Monitoring hub, Deployment pipelines, Learn, Workspaces, and a specific item 'RR DM - ADOBuild'. The main area is titled 'Workspace settings' and has a section for 'Git integration'. It includes a link to 'View Azure DevOps account', a dropdown for 'Connect Git repository and branch', and fields for 'Organization', 'Project', 'Git repository', 'Branch', and 'Git folder'. Buttons for 'Connect and sync' and 'Cancel' are at the bottom.

When Fabric Git integration finishes exporting your workspace items, your Azure DevOps branch contains a folder for each item in your workspace:

The screenshot shows the Azure DevOps interface. The top navigation bar includes 'Azure DevOps', 'DevMode', 'Repos', 'Files', and 'Demo-ADOBuild'. The left sidebar has 'DevMode' selected, along with 'Overview', 'Boards', 'Repos', 'Files', and 'Commits'. The main pane shows a repository named 'Demo-ADOBuild' with several items: 'Sales.Dataset', 'Sales.Report', 'Stocks.Dataset', 'Stocks.Report', and 'README.md'.

## Step 2 - Create and run an Azure DevOps pipeline

To create a new pipeline:

1. From the **Pipelines** tab of the left navigation menu, select **Create Pipeline** :

The screenshot shows the Azure DevOps interface with the 'DevMode' project selected. The left sidebar has a red box around the 'Pipelines' item under the 'Pipelines' heading. The main area features a cartoon illustration of a robot and a person working on a laptop. Below the illustration, the text 'Create your first Pipeline' is displayed, followed by a description: 'Automate your build and release processes using our wizard, and go from code to cloud-hosted within minutes.' A blue 'Create Pipeline' button is highlighted with a red box.

2. Select **Azure Repos Git** and select the first repository (the repository connected to the Fabric workspace):

The screenshot shows the 'Select' step of the pipeline creation wizard. The top navigation bar has tabs: 'Connect' (selected), 'Select', 'Configure', and 'Review'. The main area is titled 'New pipeline' and features a large bold text 'Where is your code?'. Below it, there is a list of source code options:

- Azure Repos Git** (YAML) - Free private Git repositories, pull requests, and code search. This option is highlighted with a red box.
- Bitbucket Cloud** (YAML) - Hosted by Atlassian
- GitHub** (YAML) - Home to the world's largest community of developers
- GitHub Enterprise Server** (YAML) - The self-hosted version of GitHub Enterprise
- Other Git** - Any generic Git repository
- Subversion** - Centralized version control by Apache

At the bottom, a note says 'Use the classic editor to create a pipeline without YAML.'

✓ Connect

Select

Configure

Review

New pipeline

## Select a repository

Filter by keywords

DevMode ▾ X



Demo-ADOBuild

### 3. Select Starter pipeline.

✓ Connect

✓ Select

Configure

Review

New pipeline

## Configure your pipeline



Starter pipeline

Start with a minimal pipeline that you can customize to build and deploy your code.



Existing Azure Pipelines YAML file

Select an Azure Pipelines YAML file in any branch of the repository.

The following YAML code appears in the editor:

✓ Connect ✓ Select ✓ Configure Review

New pipeline

## Review your pipeline YAML

❖ Demo-ADOBuild / azure-pipelines.yml \* ⓘ

```
1 # Starter pipeline
2 # Start with a minimal pipeline that you can customize to build and deploy your code.
3 # Add steps that build, run tests, deploy, and more:
4 # https://aka.ms/yaml
5
6 trigger:
7 - main
8
9 pool:
10   vmImage: ubuntu-latest
11
12 steps:
13 - script: echo Hello, world!
14   displayName: 'Run a one-line script'
15
16 - script: |
17   echo Add other tasks to build, test, and deploy your project.
18   echo See https://aka.ms/yaml
19   displayName: 'Run a multi-line script'
20
```

4. Copy and paste the [YAML code from the Power BI developer mode pipeline](#) into the pipeline you created:

Code Blame 110 lines (91 loc) · 4.53 KB ⚡ Code 55% faster with GitHub Copilot Raw ⌂ ⌄ ⌁ ⌂

```
1 trigger: none
2
3 pool:
4   vmimage: 'windows-latest'
5
6 stages:
7 - stage: Build
8   jobs:
9
10   - job: Build_Datasets
11     steps:
12       - checkout: self
```

← Demo-ADOBuild

```
main ▾  Demo-ADOBuild / azure-pipelines.yml

1 trigger: none
2
3 pool:
4   vmImage: 'windows-latest'
5
6 stages:
7   - stage: Build
8     jobs:
9
10    - job: Build_Datasets
11      steps:
12        - checkout: self
13        - path: 'self'
14          Settings
15            - task: PowerShell@2
16              displayName: 'Download Tabular Editor and Default Rules'
17              inputs:
18                targetType: inline
19                script: |
20                  $path = "$(Build.SourcesDirectory)"
21                  $tempPath = "$path\_temp"
22                  $toolPath = "$path\tools\TE"
23                  New-Item -ItemType Directory -Path $tempPath -ErrorAction SilentlyContinue | Out-Null
24
25                  Write-Host "Downloading Tabular Editor binaries"
26                  $downloadUrl = "https://github.com/otykier/TabularEditor/releases/download/2.18.2/TabularEditor.Portable.zip"
27                  $zipFile = "$tempPath\TabularEditor.zip"
28                  Invoke-WebRequest -Uri $downloadUrl -OutFile $zipFile
29                  Expand-Archive -Path $zipFile -DestinationPath $toolPath -Force
30
31                  Write-Host "Downloading Dataset default rules"
32                  $downloadUrl = "https://raw.githubusercontent.com/microsoft/Analysis-Services/master/BestPracticeRules/BPARules.json"
33                  Invoke-WebRequest -Uri $downloadUrl -OutFile "$tempPath\Rules-Dataset.json"
34
35          Settings
```

5. Select **Save and Run** to commit your new pipeline to the repository.

✓ Connect      ✓ Select      ✓ Configure      Review

New pipeline

## Review your pipeline YAML

Variables Save and run

Demo-ADOBuild / azure-pipelines.yml \* ⌂ Show assistant

```
trigger: none
pool:
  vmImage: 'windows-latest'
stages:
  - stage: Build
    jobs:
      - job: Build_Datasets
        steps:
          - checkout: self
          - path: 'self'
            Settings
              - task: PowerShell@2
                displayName: 'Download Tabular Editor and Default Rules'
                inputs:
                  targetType: inline
```

## Save and run

X

Saving will commit azure-pipelines.yml to the repository.

Commit message

Set up CI with Azure Pipelines

Optional extended description

Add an optional description...

- Commit directly to the main branch
- Create a new branch for this commit

Save and run

Azure DevOps runs the pipeline and starts two build jobs in parallel:



#20231003.1 • Set up CI with Azure Pipelines

Demo-ADOBUILD

## Summary

Triggered by

Repository and version

Demo-ADOBUILD

main ↴ 1dd4ee5f

Time started and elapsed

Just now

6s

## Jobs

Name

Build\_Datasets

Build\_Reports

- Build\_Datasets
  - Downloads Tabular Editor binaries.
  - Download Best Practice Analyzer [default rules](#). To customize the rules, add a *Rules-Dataset.json* to the root of the repository.
  - Cycle through all the semantic model item folders and run Tabular Editor BPA Rules.
- Build\_Reports
  - Download PBI Inspector binaries.
  - Download PBI Inspector [default rules](#). To customize the rules, add a *Rules-Report.json* to the root of the repository.
  - Cycle through all the report item folders and run Power BI Inspector rules.

When it finishes, Azure DevOps creates a report of all the warnings and errors it encountered:

Triggered by 

Repository and version

❖ Demo-ADOBUILD

⌚ main ⌁ 1dd4ee5f

Time started and elapsed

🕒 Today at 16:29

⌚ 54s

## Warnings 42

! Partition (M - Import) Sales-ddb4c40b-46fd-49ea-9a19-16e7e640a21a violates rule "[Performance] Minimize Power Query transformations"  
Build\_Datasets • Run Dataset Rules

! Partition (M - Import) About-77c21240-7751-4575-bf40-8c068bfd01cd violates rule "[Performance] Minimize Power Query transformations"  
Build\_Datasets • Run Dataset Rules

! Measure [Sales Amount (% ? LY)] violates rule "[Formatting] Percentages should be formatted with thousands separators and 1 decimal"  
Build\_Datasets • Run Dataset Rules

! Measure [Margin %] violates rule "[Formatting] Percentages should be formatted with thousands separators and 1 decimal"  
Build\_Datasets • Run Dataset Rules

! Measure [Margin % Overall] violates rule "[Formatting] Percentages should be formatted with thousands separators and 1 decimal"  
Build\_Datasets • Run Dataset Rules

! Measure [Value % (? ly)] violates rule "[Formatting] Percentages should be formatted with thousands separators and 1 decimal"  
Build\_Datasets • Run Dataset Rules

! Measure [# Customers (with Sales)] violates rule "[Formatting] Whole numbers should be formatted with thousands separators and no decimals"  
Build\_Datasets • Run Dataset Rules

! Measure [# Products (with Sales)] violates rule "[Formatting] Whole numbers should be formatted with thousands separators and no decimals"  
Build\_Datasets • Run Dataset Rules

! Measure [Sales Qty] violates rule "[Formatting] Whole numbers should be formatted with thousands separators and no decimals"  
Build\_Datasets • Run Dataset Rules

! Measure [# Sales] violates rule "[Formatting] Whole numbers should be formatted with thousands separators and no decimals"  
Build\_Datasets • Run Dataset Rules

[View the log to see the remaining 24 warnings for this task](#)

Select on the link to open a more detailed view of the two jobs:

! Measure [# Sales] violates rule "[Formatting] Whole numbers should be formatted with thousands separators and no decimals"  
Build\_Datasets • Run Dataset Rules

[View the log to see the remaining 24 warnings for this task](#)

← Jobs in run #20231003.1

Demo-ADOBUILD

	Build	Run Report Rules
Build		1 Starting: Run Report Rules
Build_Datasets	19s	2 =====
Initialize.job	1s	3 Task : PowerShell
Checkout Demo-ADOBUILD@main...	5s	4 Description : Run a PowerShell script on Linux, macOS, or Windows
Download Tabular Editor and Defa...	3s	5 Version : 2.228.1
Run Dataset Rules	8s	6 Author : Microsoft Corporation
Post-job: Checkout Demo-ADOB...	<1s	7 Help : <a href="https://docs.microsoft.com/azure/devops/pipelines/tasks/utility/powershell">https://docs.microsoft.com/azure/devops/pipelines/tasks/utility/powershell</a>
Finalize Job	<1s	8 =====
Build_Reports	15s	9 Generating script.
Initialize job	<1s	10 ===== Starting Command Output =====
Checkout Demo-ADOBUILD@main...	4s	11 "C:\Windows\system32\WindowsPowerShell\v1.0\powershell.exe" -NoLogo -NoProfile -NonInteractive -ExecutionPolicy
Download PBIXInspector	3s	12 Running default downloaded rules
Run Report Rules	5s	13 Processing 'D:\a\1\self\Sales.Report'
Post-job: Checkout Demo-ADOB...	<1s	14 ##[warning]"Sales" - Rule "Reduce the number of visible visuals on the page" FAILED with result: false, expect
Finalize Job	<1s	15 ###[warning]"Dynamic Measure" - Rule "Reduce the number of visible visuals on the page" FAILED with result: fal
Finalize build		16 ###[warning]"Sales" - Rule "Reduce the number of objects within visuals" FAILED with result: [
Report build status	<1s	17 "484fbdd73143c5bf71fa",
		18 "5acb1caf298449a8acb4"
		19 ], expected: [].
		20 ###[warning]"Sales Geo" - Rule "Reduce the number of objects within visuals" FAILED with result: [
		21 "a64038428c095bd71739"
		22 ], expected: [].
		23 Processing 'D:\a\1\self\Stocks.Report'
		24 ###[warning]"Sales" - Rule "Reduce the number of visible visuals on the page" FAILED with result: false, expect
		25 ###[warning]"Dynamic Measure" - Rule "Reduce the number of visible visuals on the page" FAILED with result: fal
		26 ###[warning]"Sales" - Rule "Reduce the number of objects within visuals" FAILED with result: [
		27 "484fbdd73143c5bf71fa",
		28 "5acb1caf298449a8acb4"
		29 ], expected: [].
		30 ###[warning]"Sales Geo" - Rule "Reduce the number of objects within visuals" FAILED with result: [
		31 "a64038428c095bd71739"
		32 ], expected: [].
		33 Finishing: Run Report Rules

If your report or semantic model fails a rule with a higher severity level, the build fails, and the error is highlighted:

Manually run by 

Repository and version

Demo-ADOBUILD  
main 5ca6528b

**Errors 1**   **Warnings 42**

 Measure [Margin %] violates rule "[DAX Expressions] Measure references should be unqualified"  
Build\_Datasets • Run Dataset Rules

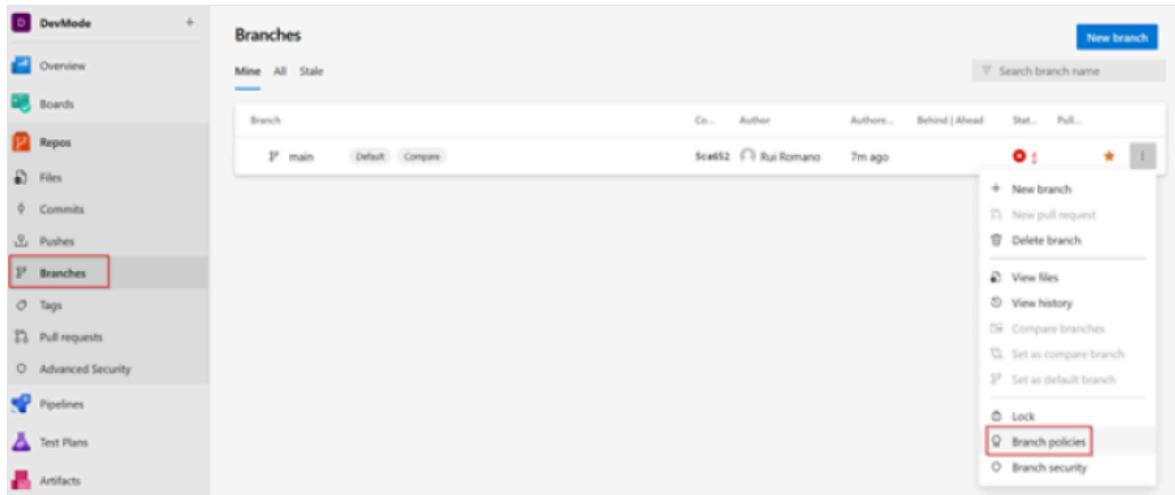
**Jobs**

Name
Build_Datasets
Build_Reports

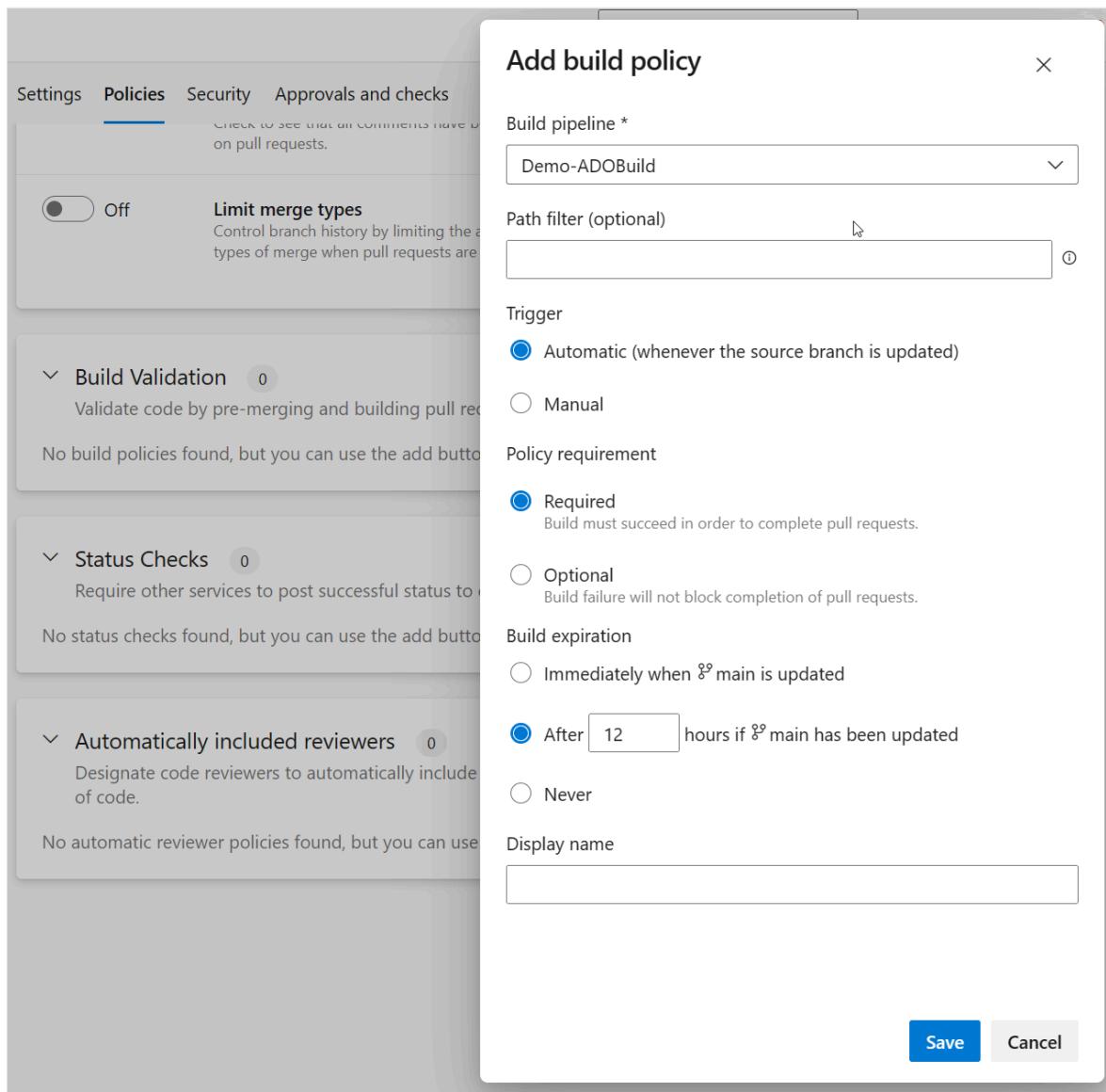
# Step 3 - Define branch policies

Once the pipeline is up and running, enable **Branch Policies** on the *main* branch. This step ensures that no commits can be made directly into main. A "[pull request](#)" is always required to merge changes back into *main* and you can configure the pipeline to run with every pull request.

1. Select **Branches > main Branch > Branch policies:**



2. Configure the created pipeline as a *Build Policy* for the branch:



**Demo-ADOBuild**

Settings Policies Security Approvals and checks

Limit merge types

Control branch history by limiting the available types of merge when pull requests are completed.

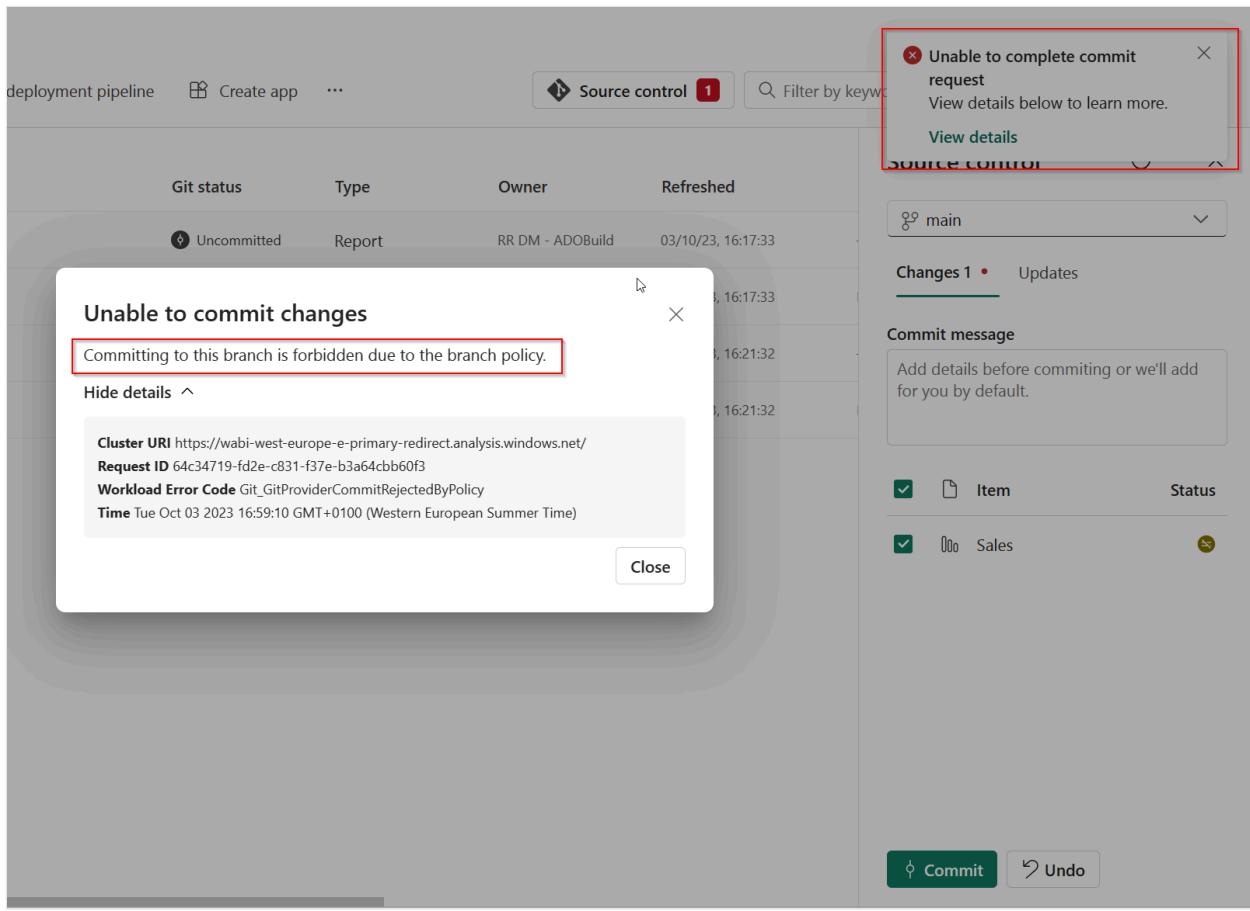
**Build Validation** 1

Validate code by pre-merging and building pull request changes.

Enabled	Name ↑	Path filter	Trigger	Inheritance
<input checked="" type="checkbox"/> On	Demo-ADOBuild		Automatic Expires after 12 hours	

## Step 4 - Create pull request

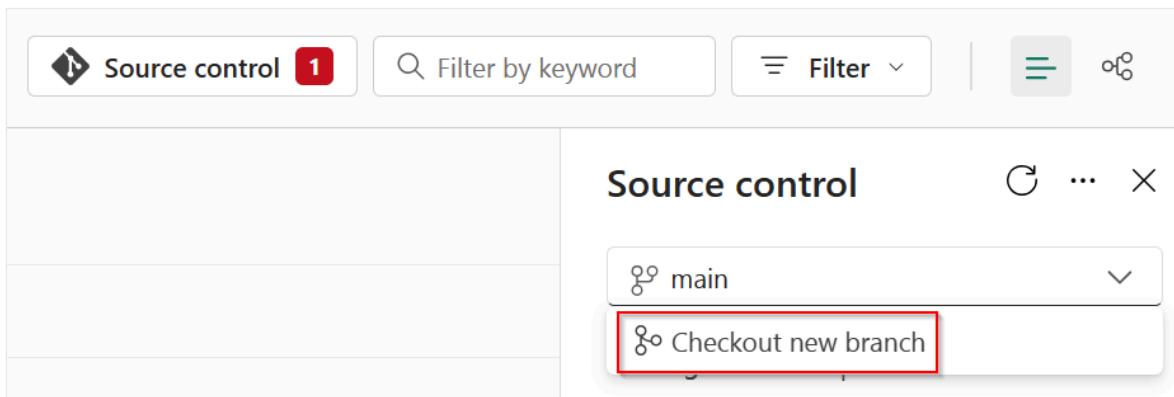
If you return to your Fabric Workspace, make a modification to one of the reports or semantic models, and attempt to commit the change, you receive the following error:



You can only make changes to the main branch through a pull request. To create a pull request checkout a new branch to make the changes on:

Create a branch directly from the Fabric Workspace:

1. In the Source Control pane, select on **Checkout new branch** and provide a name for the branch.



## Checkout branch

X

Checkout to a new branch created from the current branch.

You can commit your workspace changes to the new branch.

**Branch name \***

feature/reportChanges

**Checkout branch**

**Cancel**

Alternatively, you can choose to develop within a separate, isolated workspace or in Power BI Desktop. For more information, see [Manage Git branches](#)

2. Commit your changes to this new branch.

The screenshot shows the Power BI Source Control interface for the 'RR DM - ADOBuild' workspace. The 'Source control' dropdown is set to 'feature/reportChanges'. The commit message field contains 'report changed'. The 'Commit' button at the bottom left is highlighted with a red box. The status bar at the bottom shows the branch as 'feature/reportChanges' and the last sync time as '10/3/2023 at 5:04 PM'.

Name	Git	Status
Sales	∅	
Sales	✓	
Stocks	✓	
Stocks	✓	

**Source control**

feature/reportChanges

Changes 1 • Updates

Commit message

report changed

Item Status

Sales

**Commit**

Last synced: 10/3/2023 at 5:04 PM 5ca6528b

3. Following the commit, create a pull request into the *main* branch from the Azure DevOps portal.

The screenshot shows two screenshots of the Azure DevOps interface. The top screenshot is the 'Files' page for the 'main' branch. It displays a notification: 'You updated feature/reportChanges Just now'. To the right of the notification is a button labeled 'Create a pull request', which is highlighted with a red box. The bottom screenshot is the 'New pull request' dialog. It shows the source branch as 'feature/reportChanges' and the target branch as 'main'. The 'Title' field contains 'report change'. The 'Description' field contains 'report change'. Below the description is a rich text editor toolbar. The 'Reviewers' section has a placeholder 'Search users and groups to add as reviewers'. The 'Work items to link' section has a placeholder 'Search work items by ID or title'. The 'Tags' section is empty. At the bottom right of the dialog is a 'Create' button, which is also highlighted with a red box.

The pull request workflow not only allows you to validate and review the changes, but also automatically triggers the pipeline.

# report change

Active

!14

RR

proposes to merge [feature/reportChanges](#) into main

Overview Files Updates Commits Conflicts



1 required check running now

1 optional check not yet run



Demo-ADOBuild Build in progress

[View 2 checks](#)



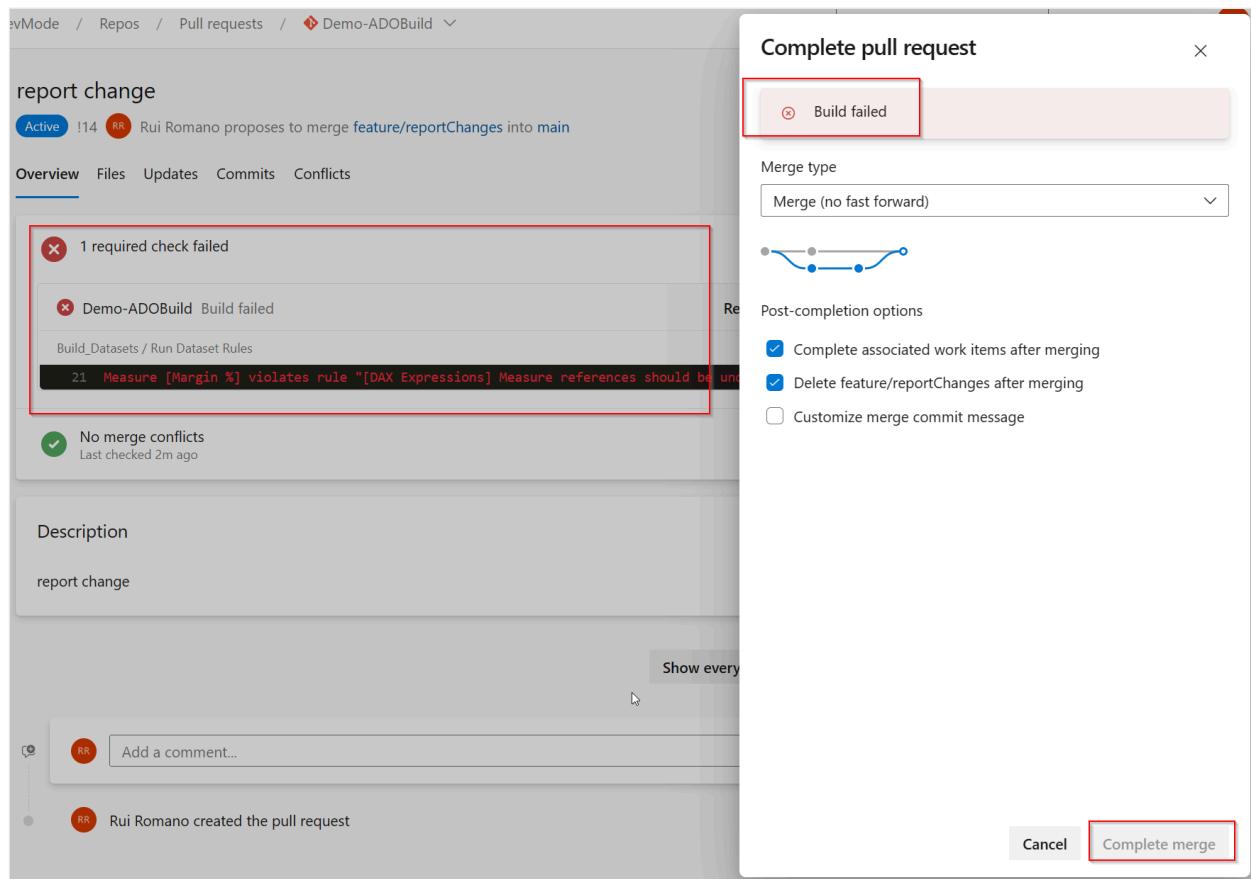
No merge conflicts

Last checked Just now

## Description

report change

If there's a high-severity error in one of the rules, you can't finalize the pull request and merge the changes back into the main branch.



## Related content

- For information about synchronizing your workspace with the Git branch, including updating your workspace and committing changes to Git, see [Get started with Git integration](#).
- For tips about different options for building CI/CD processes in Fabric based on common customer scenarios, see [Choose the best Fabric CI/CD workflow option for you](#).

## Feedback

Was this page helpful?

Yes

No

[Provide product feedback ↗](#) | [Ask the community ↗](#)

# Deploy a Power BI project using Fabric APIs

Article • 10/10/2024

This article explains how to deploy a Power BI project (.pbip) using Fabric REST APIs and a PowerShell script. This article is written for developers who need complete control over their Power BI deployment process and should serve as an example. The example's pattern can be applied to other languages or tools capable of calling the Fabric REST APIs.

## Save your work as a Power BI project

To use the Fabric REST APIs, you need to save your work as a [Power BI project file \(.pbip\)](#).

## Understand which APIs are used

To deploy the .pbip content, use the following Fabric REST APIs:

- [List Items](#) - Lists the existing items in the workspace.
- [Create Item](#) - Creates a new item.
- [Update Item Definition](#) - Updates the item definition. This API is used in case the item already exists.

## PowerShell script

To deploy your project, use a PowerShell script. The script executes the following actions:

1. Ensures the workspace exists.
2. Creates or updates the Power BI report and semantic model in the workspace, using the Power BI Project file definitions.

The script uses the [FabricPS-PBIP](#) module, which serves as a wrapper for the Fabric APIs and handles tasks such as authentication, asynchronous calls, and metadata management for Power BI project files.

## (!) Note

The powershell module *fabricps-pbip* is open-source and Microsoft does not offer support or documentation for it.

## Script template

Use this script template to deploy your Power BI project:

PowerShell

```
# Parameters
$workspaceName = "[Workspace Name]"
$pbipSemanticModelPath = "[PBIP Path]\[Item Name].SemanticModel"
$pbipReportPath = "[PBIP Path]\[Item Name].Report"
$currentPath = (Split-Path $MyInvocation.MyCommand.Definition -Parent)
Set-Location $currentPath

# Download modules and install
New-Item -ItemType Directory -Path ".\modules" -ErrorAction SilentlyContinue
| Out-Null
@("https://raw.githubusercontent.com/microsoft/Analysis-
Services/master/pbidevmode/fabricps-pbip/FabricPS-PBIP.psm1"
, "https://raw.githubusercontent.com/microsoft/Analysis-
Services/master/pbidevmode/fabricps-pbip/FabricPS-PBIP.psd1") |% {
    Invoke-WebRequest -Uri $_ -OutFile ".\modules\$([Split-Path $_ -Leaf])"
}
if(-not (Get-Module Az.Accounts -ListAvailable)) {
    Install-Module Az.Accounts -Scope CurrentUser -Force
}
Import-Module ".\modules\FabricPS-PBIP" -Force

# Authenticate
Set-FabricAuthToken -reset

# Ensure workspace exists
$workspaceId = New-FabricWorkspace -name $workspaceName -skipErrorIfExists

# Import the semantic model and save the item id
$semanticModelImport = Import-FabricItem -workspaceId $workspaceId -path
$pbipSemanticModelPath

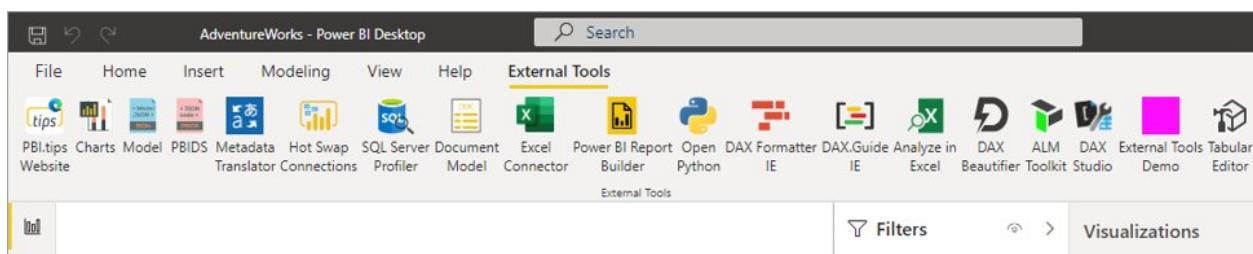
# Import the report and ensure its binded to the previous imported report
$reportImport = Import-FabricItem -workspaceId $workspaceId -path
$pbipReportPath -itemProperties @{"semanticModelId" =
$semanticModelImport.Id}
```

# External tools in Power BI Desktop

Article • 09/04/2024

Power BI has a vibrant community of business intelligence professionals and developers. Community contributors create free tools that use Power BI and Analysis Services APIs to extend and integrate with Power BI Desktop's data modeling and reporting features.

The **External Tools** ribbon provides easy access to external tools that are installed locally and *registered* with Power BI Desktop. When launched from the External Tools ribbon, Power BI Desktop passes the name and port number of its internal data model engine instance and the current model name to the tool. The tool then automatically connects, providing a seamless connection experience.



## External tool categories

External tools generally fall into one of the following categories:

**Semantic modeling** - Open-source tools such as DAX Studio, ALM Toolkit, Tabular Editor, and Metadata Translator extend Power BI Desktop functionality for specific data modeling scenarios such as Data Analysis Expressions (DAX) query and expression optimization, application lifecycle management (ALM), and metadata translation.

**Data analysis** - Tools for connecting to a model in read-only to query data and perform other analysis tasks. For example, a tool might launch Python, Excel, and Power BI Report Builder. The tool connects the client application to the model in Power BI Desktop for testing and analysis without having to first publish the Power BI Desktop (*.pbix*) file to the Power BI service. Tools to document a Power BI semantic model also fall into this category.

**Miscellaneous** - Some external tools don't connect to a model at all, but instead extend Power BI Desktop to make helpful tips and make helpful content more readily accessible. For example, PBI.tips tutorials, DAX Guide from [sqlbi.com](#), and the PowerBI.tips Product Business Ops community tool, make installation of a large selection of external tools easier. These tools also assist registration with Power BI Desktop, including DAX Studio, ALM Toolkit, Tabular Editor, and many others easy.

**Custom** - Integrate your own scripts and tools by adding a \*.pbisql.json document to the Power BI Desktop\External Tools folder.

Before installing external tools, keep the following notes in mind:

- External tools aren't supported in Power BI Desktop for Power BI Report Server.
- External tools are provided by external, third-party contributors. Except for the underlying public Microsoft APIs, Microsoft doesn't provide support or documentation for external tools. Microsoft does provide support if the issue can be reproduced with Microsoft tools. These tools include SQL Server Management Studio (SSMS), or sample code that uses the public Microsoft APIs.

## Featured open-source tools

There are many external tools out there. Here are some of the most popular and belong in every Power BI Desktop data modelers toolbox:

[ ] Expand table

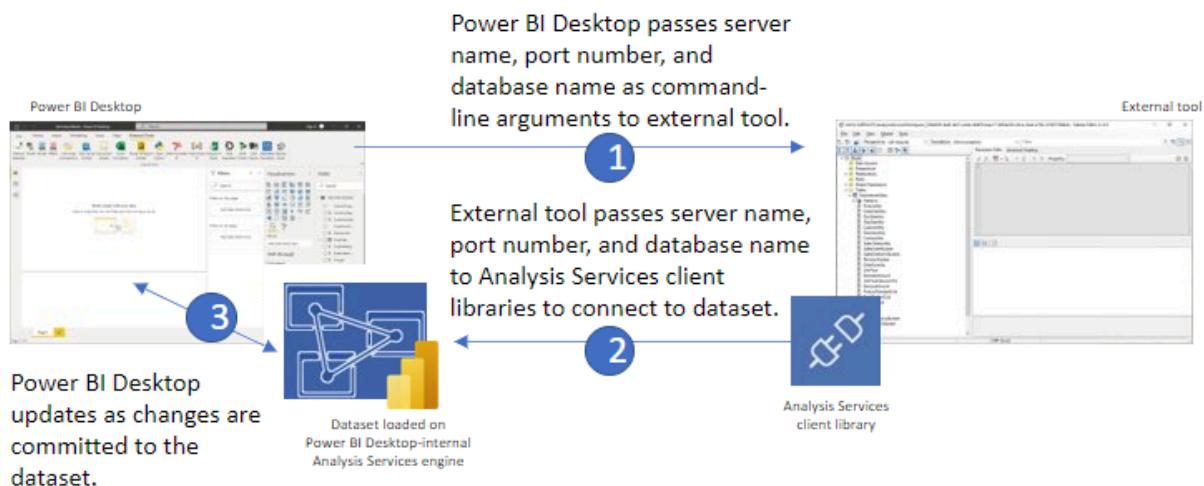
Tool	Description
PowerBI.tips - Business Ops	An easy to use deployment tool for adding external tools extensions to Power BI Desktop. The Business Ops goal is to provide a one stop shop for installing all the latest versions of external tools. To learn more, go to <a href="#">PowerBI.tips - Business Ops</a> .
Tabular Editor	Model creators can easily build, maintain, and manage tabular models by using an intuitive and lightweight editor. A hierarchical view shows all objects in your tabular model organized by display folders, with support for multi-select property editing and DAX syntax highlighting. To learn more, go to <a href="#">tabulareditor.com</a> .
DAX Studio	A feature-rich tool for DAX authoring, diagnosis, performance tuning, and analysis. Features include object browsing, integrated tracing, query execution breakdowns with detailed statistics, DAX syntax highlighting and formatting. To get the latest, go to <a href="#">DAX Studio</a> on GitHub.
ALM Toolkit	A schema compare tool for Power BI models and semantic models, used for application lifecycle management (ALM) scenarios. You can perform straightforward deployment across environments and retain incremental refresh historical data. You can diff and merge metadata files, branches, and repos. You can also reuse common definitions between semantic models. To get the latest, go to <a href="#">alm-toolkit.com</a> .
Metadata Translator	Streamlines localization of Power BI models and semantic models. The tool can automatically translate captions, descriptions, and display folder names of tables,

Tool	Description
	columns, measures, and hierarchies. The tool translates by using the machine translation technology of Azure Cognitive Services. You can also export and import translations via Comma Separated Values (.csv) files for convenient bulk editing in Excel or a localization tool. To get the latest, go to <a href="#">Metadata Translator</a> on GitHub.

## External tools integration architecture

Power BI Desktop (*pbix*) files consist of multiple components including the report canvas, visuals, model metadata, and any data that was loaded from data sources. When Power BI Desktop opens a *pbix* file, it launches an Analysis Services process in the background to load the model so that the data modeling features and report visuals can access model metadata and query model data.

When Power BI Desktop launches Analysis Services as its analytical data engine, it dynamically assigns a random port number. It also loads the model with a randomly generated name in the form of a globally unique identifier (GUID). Because these connection parameters change with every Power BI Desktop session, it's difficult for external tools to discover on their own the correct Analysis Services instance and model to connect to. External tools integration solves this problem by allowing Power BI Desktop to send the Analysis Services server name, port number, and model name to the tool as command-line parameters when starting the external tool from the External Tools ribbon, as shown in the following diagram.



With the Analysis Services Server name, port number, and model name, the tool uses Analysis Services client libraries to establish a connection to the model, retrieve metadata, and execute DAX or MDX queries. Whenever an external data modeling tool updates the metadata, Power BI Desktop synchronizes the changes so that the Power BI

Desktop user interface reflects the current state of the model accurately. Keep in mind there are some limitations to the synchronization capabilities as described later.

## Data modeling operations

External tools, which connect to Power BI Desktop's Analysis Services instance, can make changes (write operations) to the data model. Power BI Desktop then synchronizes those changes with the report canvas so they're shown in report visuals. For example, external data modeling tools can override the original format string expression of a measure, and edit any of the measure properties including KPIs and detail rows. External tools can also create new roles for object and row-level security, and add translations.

## Supported write operations

Objects that support write operations:

 Expand table

Object	Connect to AS instance
Tables	No
Columns	Yes [1]
Calculated tables	Yes
Calculated columns	Yes
Relationships	Yes
Measures	Yes
Model KPIs	Yes
Calculation groups	Yes
Perspectives	Yes
Translations	Yes
Row Level Security (RLS)	Yes
Object Level Security (OLS)	Yes
Annotations	Yes
M expressions	No

[1] When using external tools to connect to the AS instance, changing a column's data type is supported, however, renaming columns isn't supported.

Power BI Desktop *project files* offer a broader scope of supported write operations. Those objects and operations that don't support write operations by using external tools to connect to Power BI Desktop's Analysis Services instance may be supported by editing Power BI Desktop project files. To learn more, see [Power BI Desktop projects - Model authoring](#).

## Data modeling limitations

All Tabular Object Model (TOM) metadata can be accessed for read-only. Write operations are limited because Power BI Desktop must remain in-sync with the external modifications, therefore the following operations aren't supported:

- Any TOM object types not covered in Supported write operations, such as tables and columns.
- Editing a Power BI Desktop template (PBIT) file.
- Report-level or data-level translations.
- Renaming tables and columns isn't yet supported
- Sending processing commands to a semantic model loaded in Power BI Desktop

## Registering external tools

External tools are *registered* with Power BI Desktop when the tool includes a \*.pbitool.json registration file in the `C:\Program Files (x86)\Common Files\Microsoft Shared\Power BI Desktop\External Tools` folder. When a tool is registered, and includes an icon, the tool appears in the External Tools ribbon. Some tools, like ALM Toolkit and DAX Studio create the registration file automatically when you install the tool. However, many tools, like SQL Profiler typically don't because the installer they do have doesn't include creating a registration file for Power BI Desktop. Tools that don't automatically register with Power BI Desktop can be registered manually by creating a \*.pbitool.json registration file.

To learn more, including json examples, see [Register an external tool](#).

## Disabling the External Tools ribbon

The External Tools ribbon is enabled by default, but can be disabled by using Group Policy or editing the `EnableExternalTools` registry key directly.

- Registry key: `Software\Policies\Microsoft\Power BI Desktop\`
- Registry value: `EnableExternalTools`

A value of 1 (decimal) enables the External Tools ribbon, which is also the default value.

A value of 0 (decimal) disable the External Tools ribbon.

## Related content

- [Register an external tool](#)
- 

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Ask the community ↗](#)

# Register an external tool

Article • 01/15/2024

Some tools must be manually registered with Power BI Desktop. To register an external tool, create a JSON file with the following example code:

```
JSON

{
    "name": "<tool name>",
    "description": "<tool description>",
    "path": "<tool executable path>",
    "arguments": "<optional command line arguments>",
    "iconData": "image/png;base64,<encoded png icon data>"
}
```

The pbitool.json file includes the following elements:

- **name:** Provide a name for the tool, which will appear as a button caption in the External Tools ribbon within Power BI Desktop.
- **description:** (optional) Provide a description, which will appear as a tooltip on the External Tools ribbon button within Power BI Desktop.
- **path:** Provide the fully qualified path to the tool executable.
- **arguments:** (optional) Provide a string of command-line arguments that the tool executable should be launched with. You can use any of the following placeholders:
  - **%server%:** Replaced with the server name and portnumber of the local instance of Analysis Services Tabular for imported/DirectQuery data models.
  - **%database%:** Replaced with the database name of the model hosted in the local instance of Analysis Services Tabular for imported/DirectQuery data models.
- **iconData:** Provide image data, which will be rendered as a button icon in the External Tools ribbon within Power BI Desktop. The string should be formatted according to the syntax for Data URIs without the "data:" prefix.

Name the file "`<tool name>.pbitool.json`" and place it in the following folder:

- `%commonprogramfiles%\Microsoft Shared\Power BI Desktop\External Tools`

For 64-bit environments, place the files in the following folder:

- `Program Files (x86)\Common Files\Microsoft Shared\Power BI Desktop\External Tools`

Files in that specified location with the **.pbitool.json** extension are loaded by Power BI Desktop upon startup.

## Example

The following **\*.pbitool.json** file launches `powershell.exe` from the External Tools ribbon and runs a script called `pbiToolsDemo.ps1`. The script passes the server name and port number in the `-Server` parameter and the semantic model name in the `-Database` parameter.

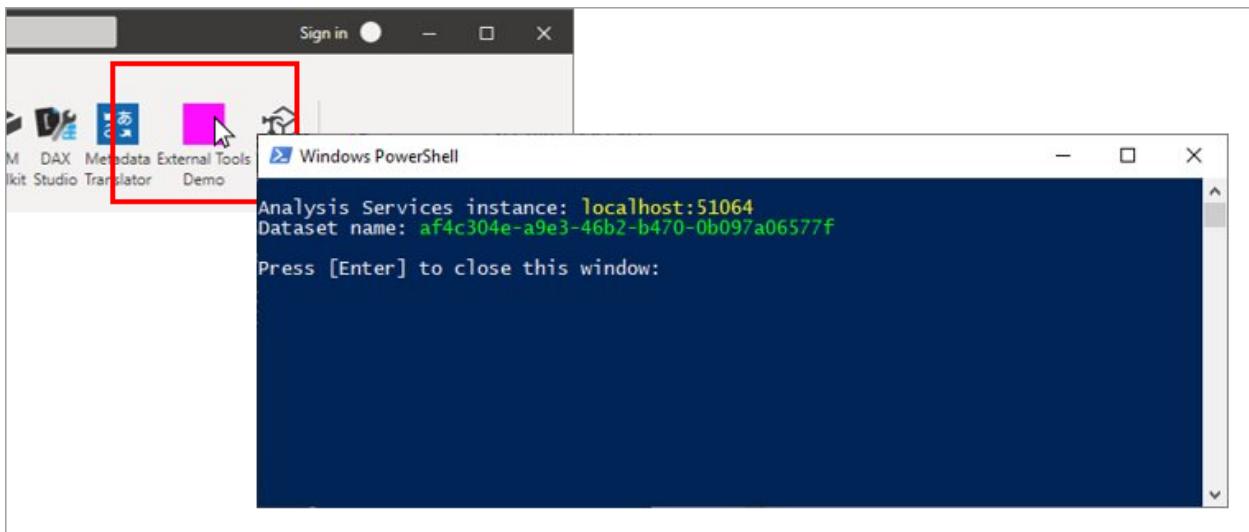
```
JSON

{
    "version": "1.0.0",
    "name": "External Tools Demo",
    "description": "Launches PowerShell and runs a script that outputs server and database parameters. (Requires elevated PowerShell permissions.)",
    "path": "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe",
    "arguments": "C:\\pbiToolsDemo.ps1 -Server \"%server%\" -Database \"%database%\"",
    "iconData": "image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAAffFcSJAAAAAXNSR0IAr4c6QAAAARnQU1BAACxjwv8YQUAAAJcEhZcwAADsEAAA7BAbiRa+0AAAANSURBVhXY/jH9+8/AAciAwpql7QkAAAAAE1FTkSuQmCC"
}
```

The corresponding `pbiToolsDemo.ps1` script outputs the Server and Database parameters to the console.

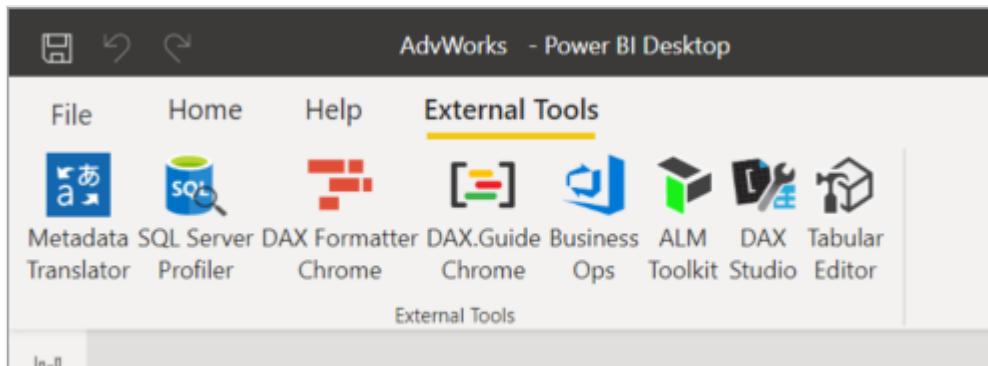
```
PowerShell

[CmdletBinding()]
param
(
    [Parameter(Mandatory = $true)]
    [string] $Server,
    [Parameter(Mandatory = $true)]
    [string] $Database
)
Write-Host ""
Write-Host "Analysis Services instance: " -NoNewline
Write-Host "$Server" -ForegroundColor Yellow
Write-Host "Dataset name: " -NoNewline
Write-Host "$Database" -ForegroundColor Green
Write-Host ""
Read-Host -Prompt 'Press [Enter] to close this window'
```



## Icon data URIs

To include an icon in the External Tools ribbon, the pbitool.json registration file must include an iconData element.



The iconData element takes a data URI without the **data:** prefix. For example, the data URI of a one pixel magenta png image is:

```
data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAFcSJAAAAAXNSR0IArs4c6
QAAAARnQU1BAACxjwv8YQUAAAJcEhZcwAADsEAAA7BAbiRa+0AAAANSURBVbXY/jH9+8/AAcAwpq17Qk
AAAAAE1FTkSuQmCC
```

Be sure to remove the **data:** prefix, as shown in the pbitool.json preceding example.

To convert a .png or other image file type to a data URI, use an online tool or a custom tool such as the one shown in the following C# code snippet:

```
c#
string ImageDataUri;
OpenFileDialog openFileDialog1 = new OpenFileDialog();
openFileDialog1.Filter = "PNG Files (.png)|*.png|All Files (*.*)|*.*";
openFileDialog1.FilterIndex = 1;
openFileDialog1.Multiselect = false;
```

```
openFileDialog1.CheckFileExists = true;  
bool? userClickedOK = openFileDialog1.ShowDialog();  
if (userClickedOK == true)  
{  
    var fileName = openFileDialog1.FileName;  
    var sb = new StringBuilder();  
    sb.Append("image/")  
        .Append((System.IO.Path.GetExtension(fileName) ??  
"png").Replace(".", ""))  
        .Append(";base64,")  
        .Append(Convert.ToBase64String(File.ReadAllBytes(fileName)));  
    ImageDataUri = sb.ToString();  
}
```

## Related content

- [External tools in Power BI Desktop](#)
- [Analysis Services client libraries](#)
- [Tabular Object Model \(TOM\)](#)