Enforcing Power BI Best Practices with Azure DevOps

Dhyanendra Singh Rathore

JAN 2025

Dhyanendra Singh Rathore

Power BI Tech Lead @ Autoliv

- ✓ Stockholm, Sweden
- ✓ Tech Blogger & Speaker
- ✓ Automation & Optimization
- ✓ Anime & Hiking

in dhyans

bits2BI

Autoliv

THANK YOU

Data Community Austria Day 2025

Awesome Partner

Microsoft

Platinum

redgate

ACP IT for innovators.

Gold

b.telligent smart data. smart decisions.

Lucient

Bronze

Tabular Editor

paiqo The Platform & AI Company

Power BI Camp www.linearis.at

Agenda

- Overview of best practices
- Best Practices today
- Enforcing best practices with Azure DevOps
- Demo
- Benefits & limitations
- Q & A

What are best practices?

Best practices refers to established techniques, methods, or processes that are considered most effective in delivering optimal results in a particular field or activity.

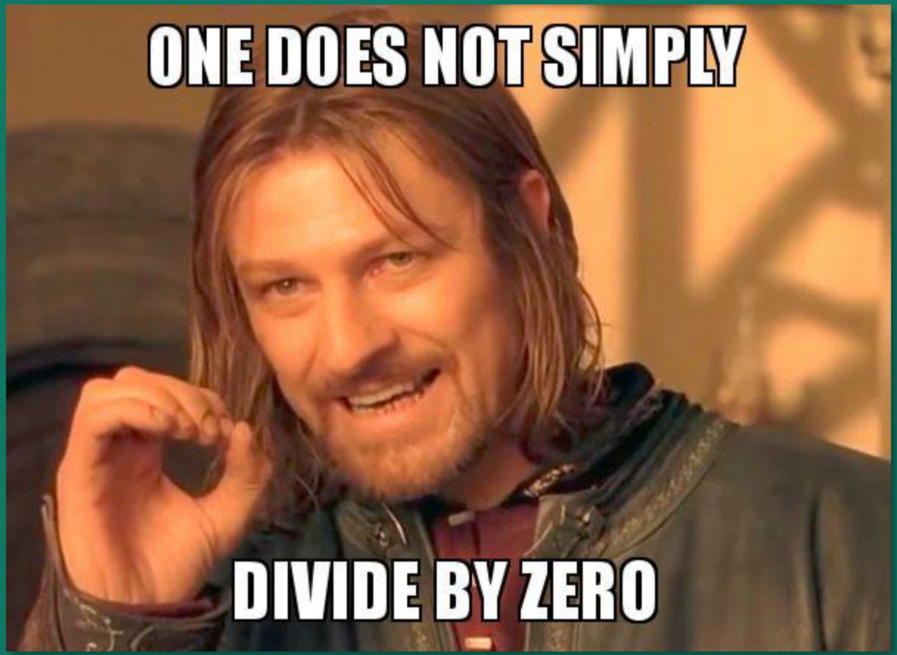
Best practices are typically based on research, experience, and lessons learned from both success and failure over time.

A simple example

Calculate Sales Margin %

Margin % = (Sales - Cost) / Sales * 100

~~Margin % = [Margin] / [Sales]~~



A simple example

Calculate Sales Margin %
Margin % = (Sales – Cost) / Sales * 100

1 Margin % = [Margin] / [Sales]

IF ([Sales] > 0, [Margin] / [Sales], *BLANK ()*) 2

IFERROR ([Margin] / [Sales], BLANK ()) 3

DIVIDE ([Margin], [Sales], *BLANK ()*) 4

Average execution time (milli seconds)

	2 M rows	21 M rows
1	61	78
2	76	143
3	89	142
4	65	79

Why are they important?

Enhanced quality: consistent outcomes & improved accuracy

Risk Mitigation: avoiding common pitfalls

Faster problem-solving: reduced learning curve

Time and cost savings

Performance and efficiency

Best practices today

- Best practices have evolved and **codified** as JSON rules
- Open-source community tools are available to evaluate **semantic models** and **reports** for best practice violations

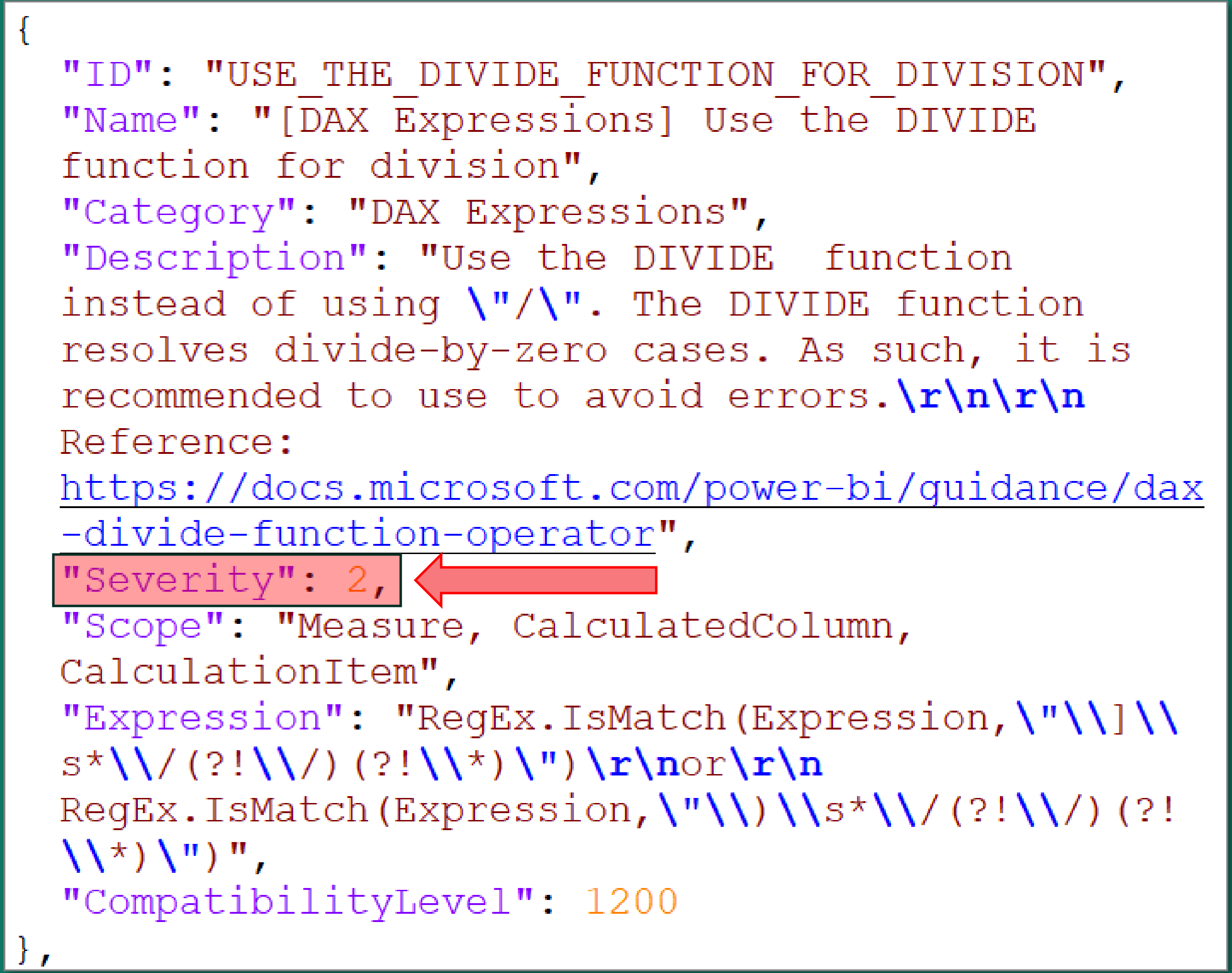
Semantic Model

Best Practice Analyzer (PBA) within Tabular Editor 2

Report

PBI-Inspector

Anatomy of JSON rules: Semantic model

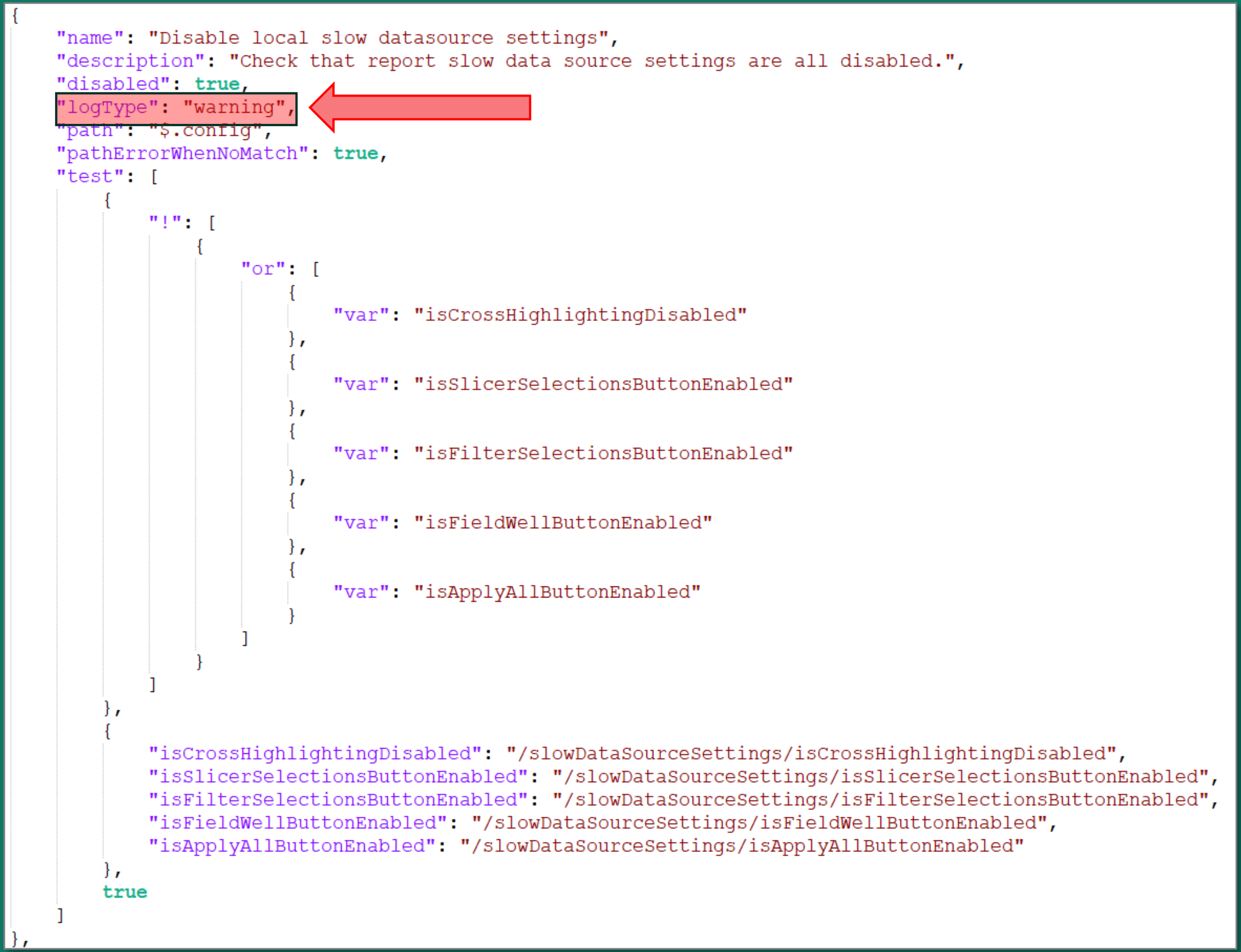


```
{
  "ID": "USE_THE_DIVIDE_FUNCTION_FOR_DIVISION",
  "Name": "[DAX Expressions] Use the DIVIDE
  function for division",
  "Category": "DAX Expressions",
  "Description": "Use the DIVIDE  function
  instead of using \"/\". The DIVIDE function
  resolves divide-by-zero cases. As such, it is
  recommended to use to avoid errors.\r\n\r\n
  Reference:
  https://docs.microsoft.com/power-bi/guidance/dax
  -divide-function-operator",
  "Severity": 2,
  "Scope": "Measure, CalculatedColumn,
  CalculationItem",
  "Expression": "RegEx.IsMatch(Expression,\"\\]\\
  s*\\/(?!\\/)(?!\\*)\")\r\nor\r\n
  RegEx.IsMatch(Expression,\"\\)\\s*\\/(?!\\/)(?!
  \\*)\")",
  "CompatibilityLevel": 1200
},
```

Severity

1. Informational
2. Warning
3. Error

Anatomy of JSON rules: Report



```
{
    "name": "Disable local slow datasource settings",
    "description": "Check that report slow data source settings are all disabled.",
    "disabled": true,
    "logType": "warning",
    "path": "$.config",
    "pathErrorWhenNoMatch": true,
    "test": [
        {
            "!": [
                {
                    "or": [
                        {
                            "var": "isCrossHighlightingDisabled"
                        },
                        {
                            "var": "isSlicerSelectionsButtonEnabled"
                        },
                        {
                            "var": "isFilterSelectionsButtonEnabled"
                        },
                        {
                            "var": "isFieldWellButtonEnabled"
                        },
                        {
                            "var": "isApplyAllButtonEnabled"
                        }
                    ]
                }
            ]
        },
        {
            "isCrossHighlightingDisabled": "/slowDataSourceSettings/isCrossHighlightingDisabled",
            "isSlicerSelectionsButtonEnabled": "/slowDataSourceSettings/isSlicerSelectionsButtonEnabled",
            "isFilterSelectionsButtonEnabled": "/slowDataSourceSettings/isFilterSelectionsButtonEnabled",
            "isFieldWellButtonEnabled": "/slowDataSourceSettings/isFieldWellButtonEnabled",
            "isApplyAllButtonEnabled": "/slowDataSourceSettings/isApplyAllButtonEnabled"
        },
        true
    ]
},
```

logType

- warning
- error

The challenge

Best practices are NOT enforced and relies on the discretion of the developers

How can we enforce the Best Practices in our development lifecycle?

Prerequisites

Power BI/Fabric

✓ Power BI Pro license
✓ Power BI Premium

OR

✓ Fabric Capacity

Azure DevOps

✓ Active account & license
✓ Access to a repository

OR

✓ Rights to create a repository

Admin Portal: Tenant settings

✓ Users can synchronize workspace items with their Git repositories

DevOps – Relevant Practices

Version Control

Continuous integration (CI)

Continuous delivery (CD)

DevOps – Relevant Terms

Branches

Branch Policies

Pull Requests (PR)

Build Pipelines

Enforcing best practices with Azure DevOps

1. Connect workspace to Azure DevOps
2. Create an Azure DevOps pipeline
3. Define branch policies
4. Create a pull request

Demo setup

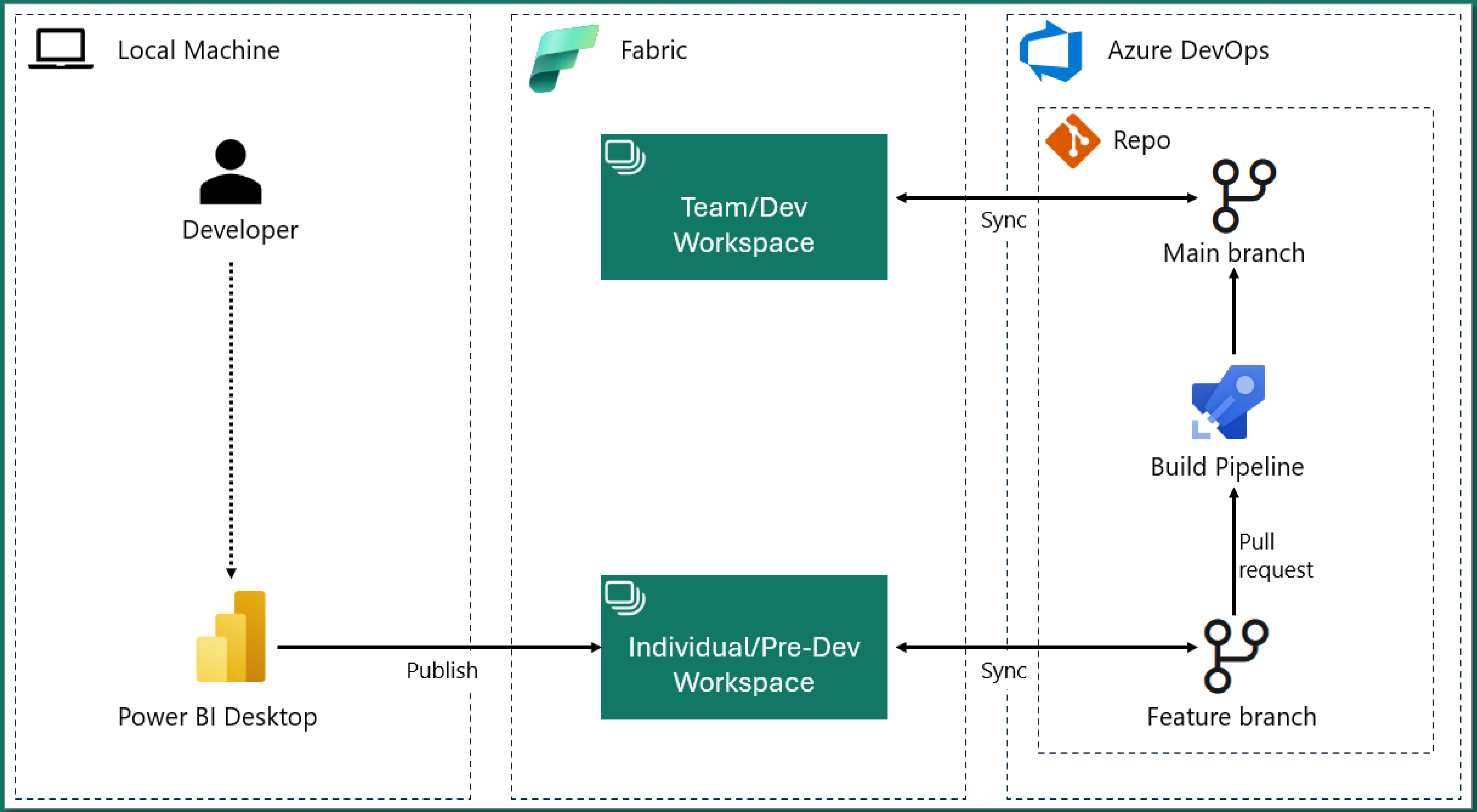
Sample Contoso PBIX from our Italian friends

Azure DevOps Repository

Power BI Workspaces

- **Pre-DEV:** Team's dev workspace
- **DEV:** Team's dev workspace with best practices enforced models and reports

Demo workflow



Demo Time!!

Benefits & Limitations

- Guaranteed best practices within a workspace
- Automated advanced quality tests with Python
- Enhanced team efficiency
- Independent of development method
- No extra cost/license*
- Customize best practices to suit your needs

- Advanced best practices that utilizes Vertipaq Analyzer can't be enforced

Key takeaways

Avoid the common pitfalls with best practices

Customize best practices to suit your needs

Enforce best practices with Azure DevOps

bits2BI

Questions?

Feedback, please :)

Thank You!!

Dhyanendra Singh Rathore

in dhyans

bits2BI