Enforcing Power BI Best Practices with Azure DevOps

Dhyanendra Singh Rathore

Agenda

- 👁 **Overview of best practices**

- 🏅 **Best Practices today**

- 🔷 **Enforcing best practices with Azure DevOps**

- 👓 **Demo**

- 🐷 **Benefits & limitations**

- ❓ **Q & A**

bits2BI

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.
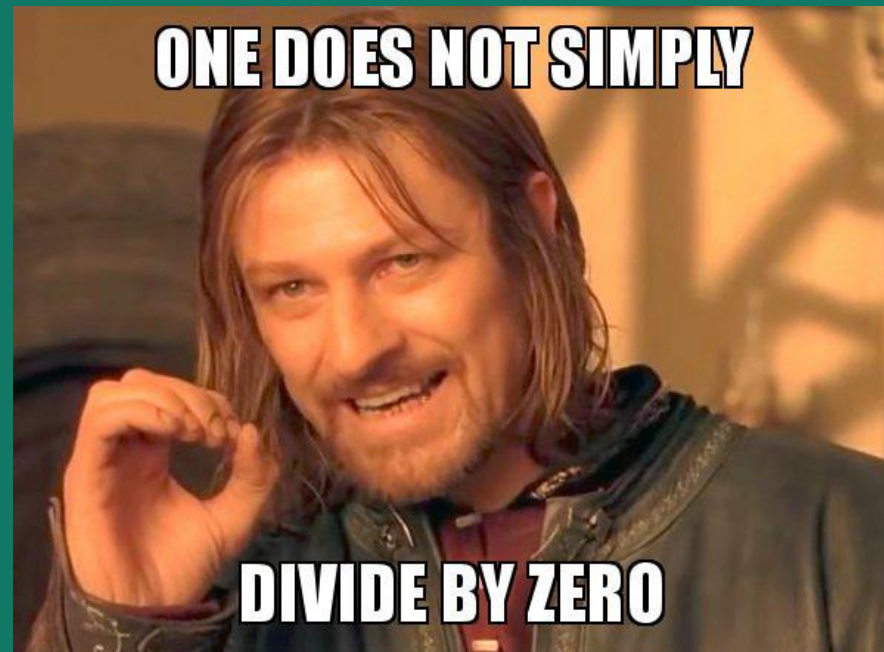
bits2BI

A simple example

Calculate Sales Margin %

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

Margin % = [Margin] / [Sales]

ONE DOES NOT SIMPLY

DIVIDE BY ZERO

A simple example

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

1 Margin % = [Margin] / [Sales]

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

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

4 DIVIDE ([Margin], [Sales], *BLANK ()*)
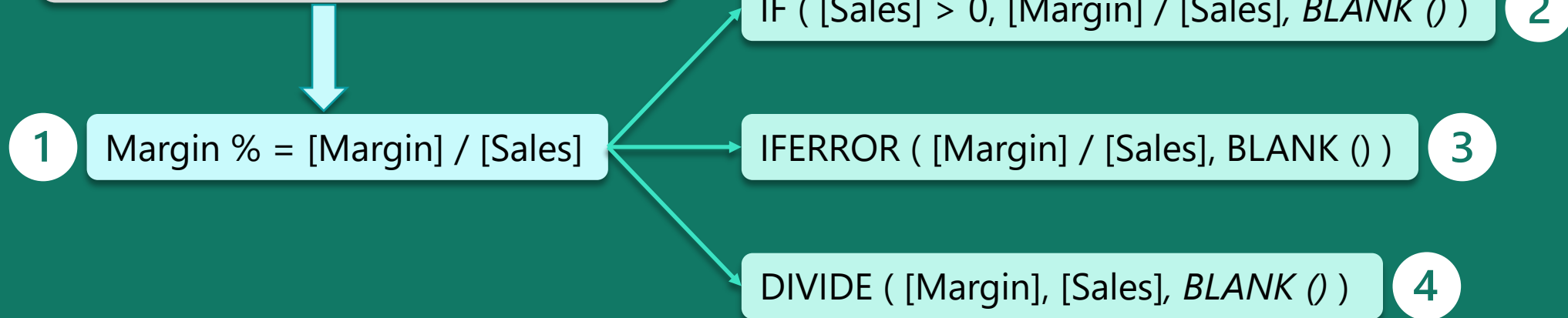
Average execution time (milli seconds)

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

🏆

bits2BI

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

bits2BI

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

```json
{
    "ID": "DAX_DIVISION_COLUMNS",
    "Name": "Avoid division (use DIVIDE function instead)",
    "Category": "DAX Expressions",
    "Description": "Calculated Columns, Measures or
    Calculated Tables should not use the division symbol in
    their expressions (/) unless the denominator is a
    constant value. Instead, it is advised to always use the
    DIVIDE(<numerator>,<denominator>) function.",
    "Severity": 3,
    "Scope": "Measure, CalculatedColumn, CalculatedTable",
    "Expression": "Tokenize().Any(\n    Type = DIV and\n
    Next.Type <> INTEGER_LITERAL and\n    Next.Type <>
    REAL_LITERAL\n)",
    "CompatibilityLevel": 1200,
    "Source": "standard\\DAX Expressions"
}
```

Severity
1. Informational
2. Warning
3. Error

bits2BI

Anatomy of JSON rules: Report

```json
{
    "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
    ]
},
```
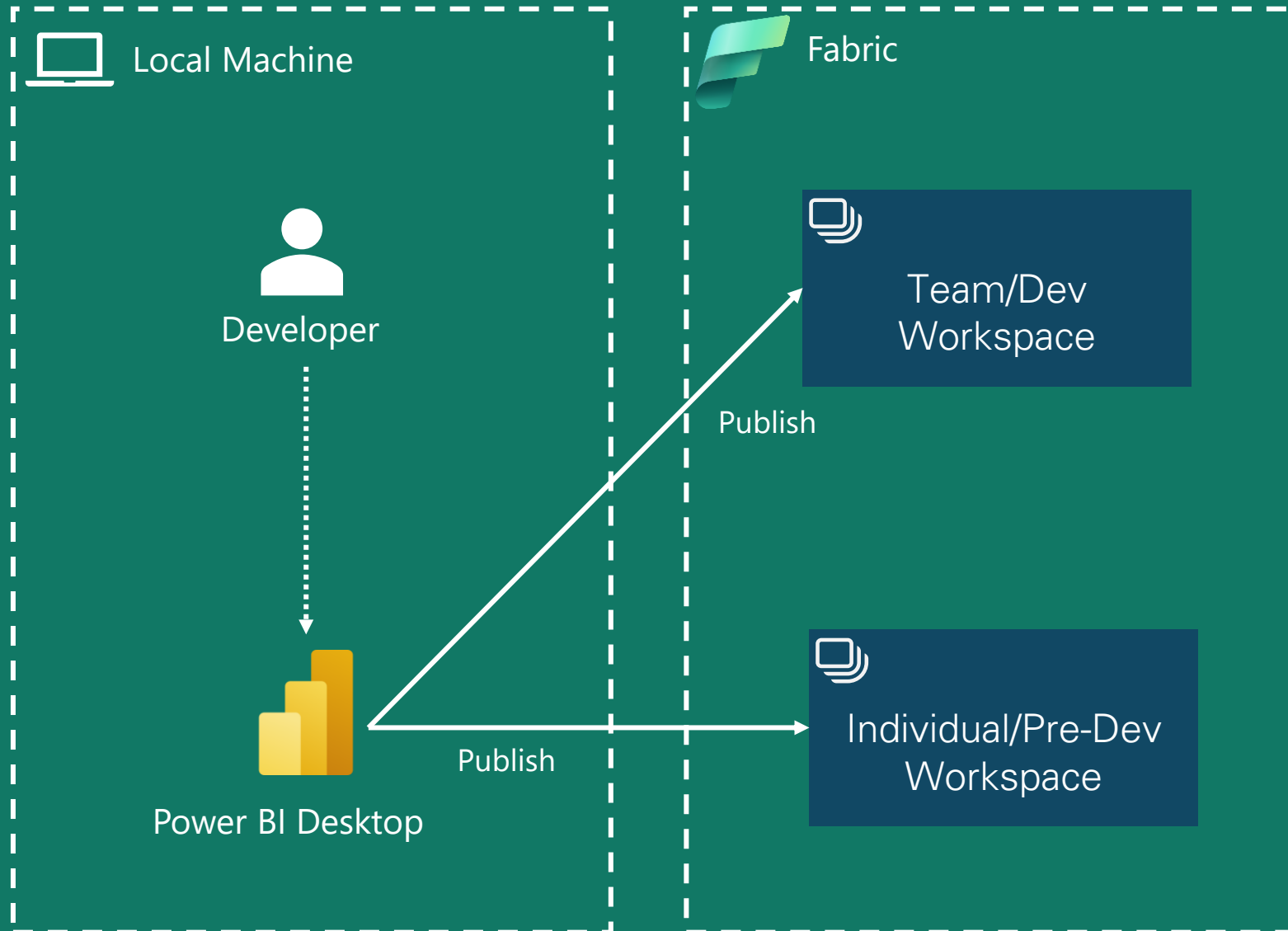
logType
- warning
- error

bits2BI

The challenge

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

bits2BI

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

bits2BI

Traditional Development Workflow

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
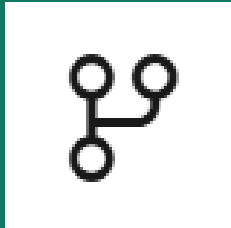
bits2BI

DevOps – Relevant Practices

Version Control
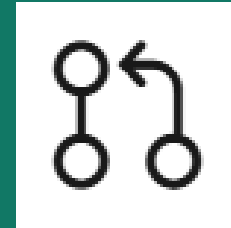

Continuous integration (CI)

Continuous delivery (CD)

DevOps – Relevant Terms
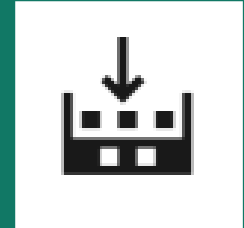
Branches

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
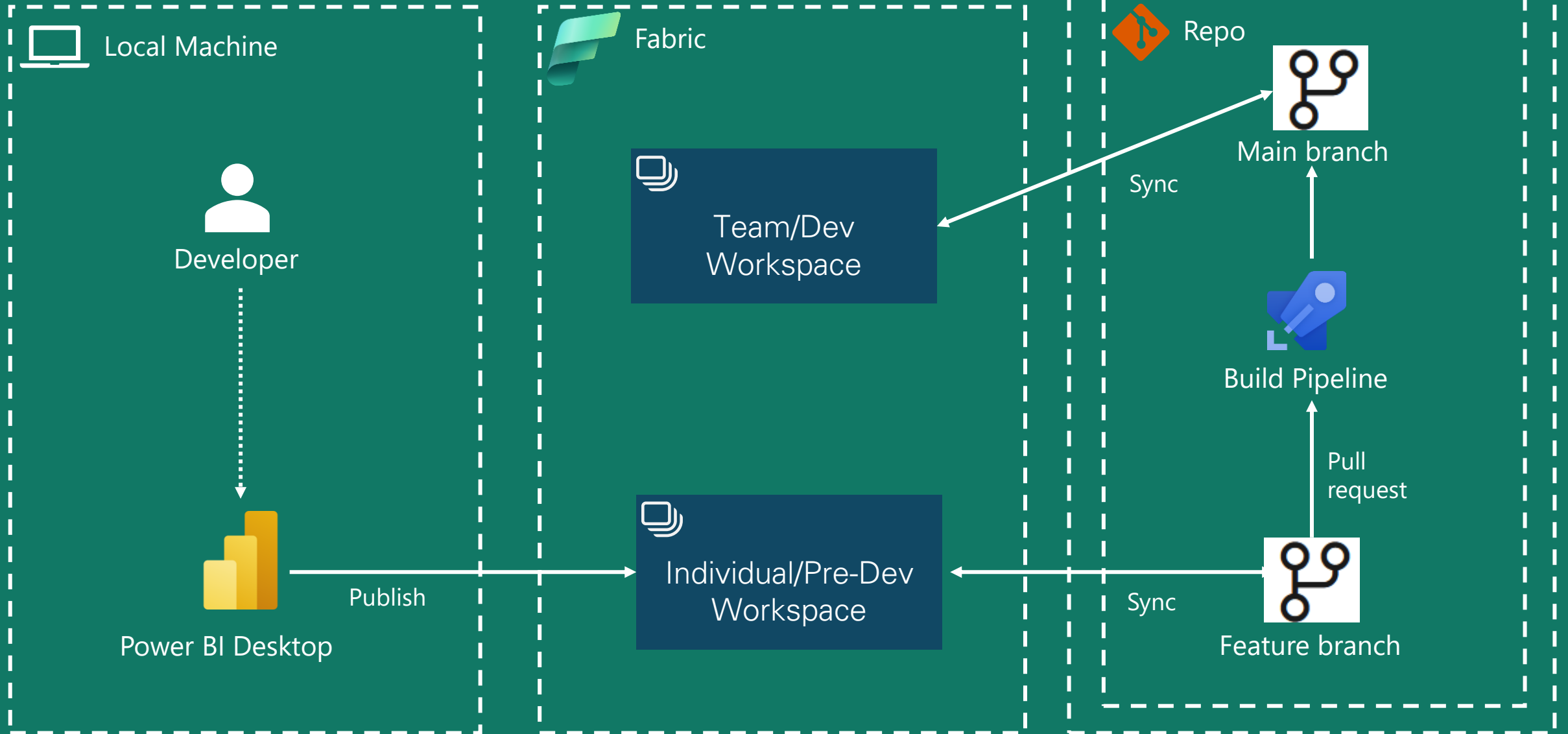
3 Define branch policies

4 Create a pull request

bits2BI

Demo Time!!

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 evaluated models and reports

bits2BI

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
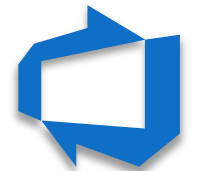
bits2BI

Takeaways

Avoid the common pitfalls with best practices

Customize best practices to suit your needs

Enforce best practices with Azure DevOps

bits2BI

Questions?

Get step-by-step guide to implement this feature at bits2bi.com

Thank You!!

Dhyanendra Singh Rathore

in **dhyans**

bits2BI