DevOps Thunder: Crushing Power BI CI/CD with Azure DevOps

Dhyanendra Singh Rathore

Power BI Tech Lead @ Autoliv

- ✓ **Stockholm, Sweden**
- ✓ **Blogger & Speaker**
- ✓ **Automation, Optimization & Mushrooms**

dhyans

bits2BI

Agenda: Find Your Sound, Don’t Copy Someone Else’s

Bit of DevOps Overview

CI strategies

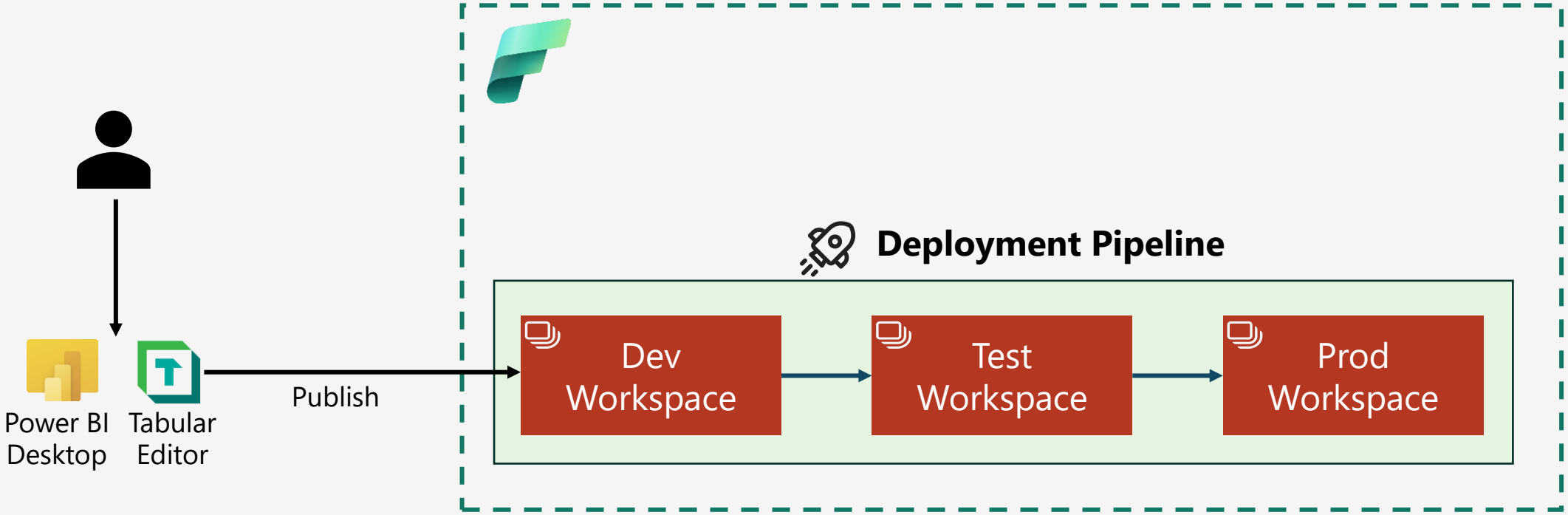
CD strategies

Comparing CI/CD strategies

Find Your Sound

Q & A

Old School Power BI Workflow

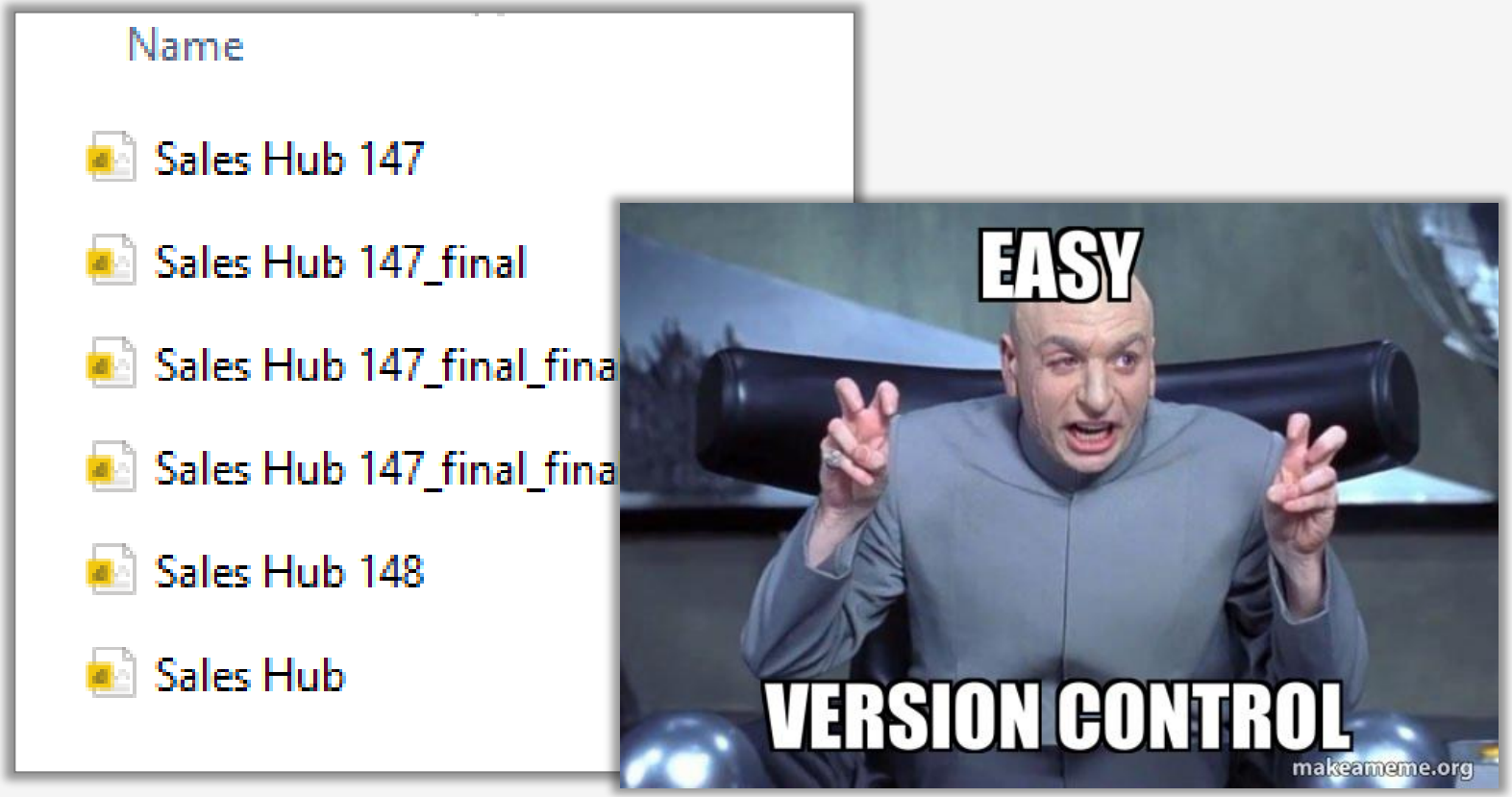


Chaos

Change history

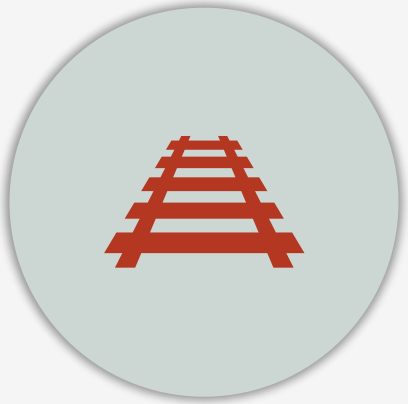
Parallel development

Harmony


Name
Sales Hub 147
Sales Hub 147_final
Sales Hub 147_final_fina
Sales Hub 147_final_fina
Sales Hub 148
Sales Hub
EASY
VERSION CONTROL
makeameme.org

Why DevOps?

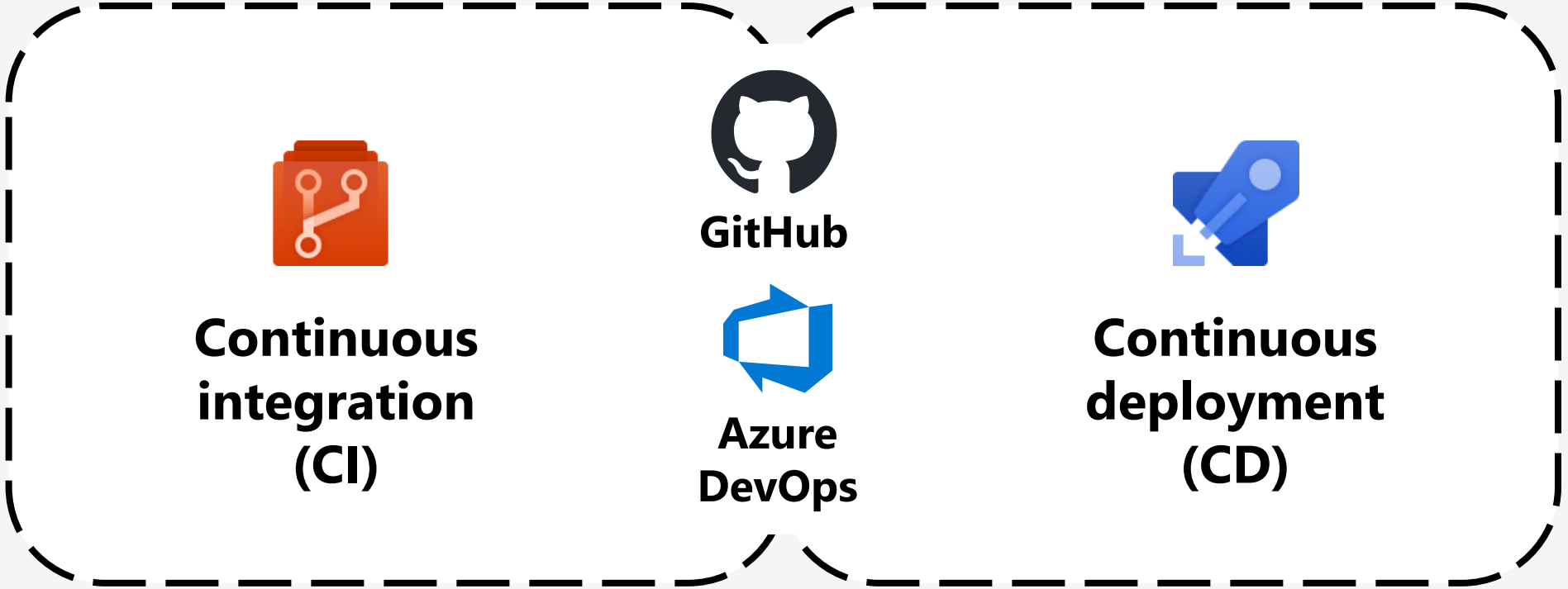
Change history



Parallel development

Automated quality checks

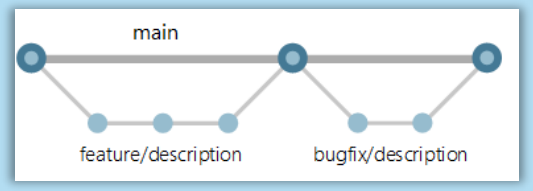
DevOps – CI/CD


Continuous integration (CI)
GitHub
Azure DevOps
Continuous deployment (CD)

Branching Strategy – Ground Rules

Use feature branches for your work

- Use feature branches for new features and bug fixes
- Use a naming convention for feature branches



Review and merge code with pull requests

- Avoid merging branches to the main branch without a pull request
- 2 reviewers is an optimal number
- Share reviewer responsibilities across the team

Keep a high quality, up-to-date main branch

- Automatically add reviewers when a pull request is created
- Setup automatic best practice evaluation and other quality checks with build pipelines

NO BODY COMMITS TO THE MAIN BRANCH

Prerequisites

Power BI/Fabric

- ✓ Power BI Pro license
- ✓ Power BI Premium
  OR
- ✓ Fabric Capacity
- ✓ Workspace Admin

Azure DevOps

- ✓ Access or rights to a create repository
- ✓ Rights to create build & release pipelines

Admin Portal: Tenant settings

- ✓ Users can synchronize workspace items with their Git repositories

bits2BI

Disclaimer

Git integration is in PREVIEW

Continuous Integration (CI)

Defining a CI Strategy

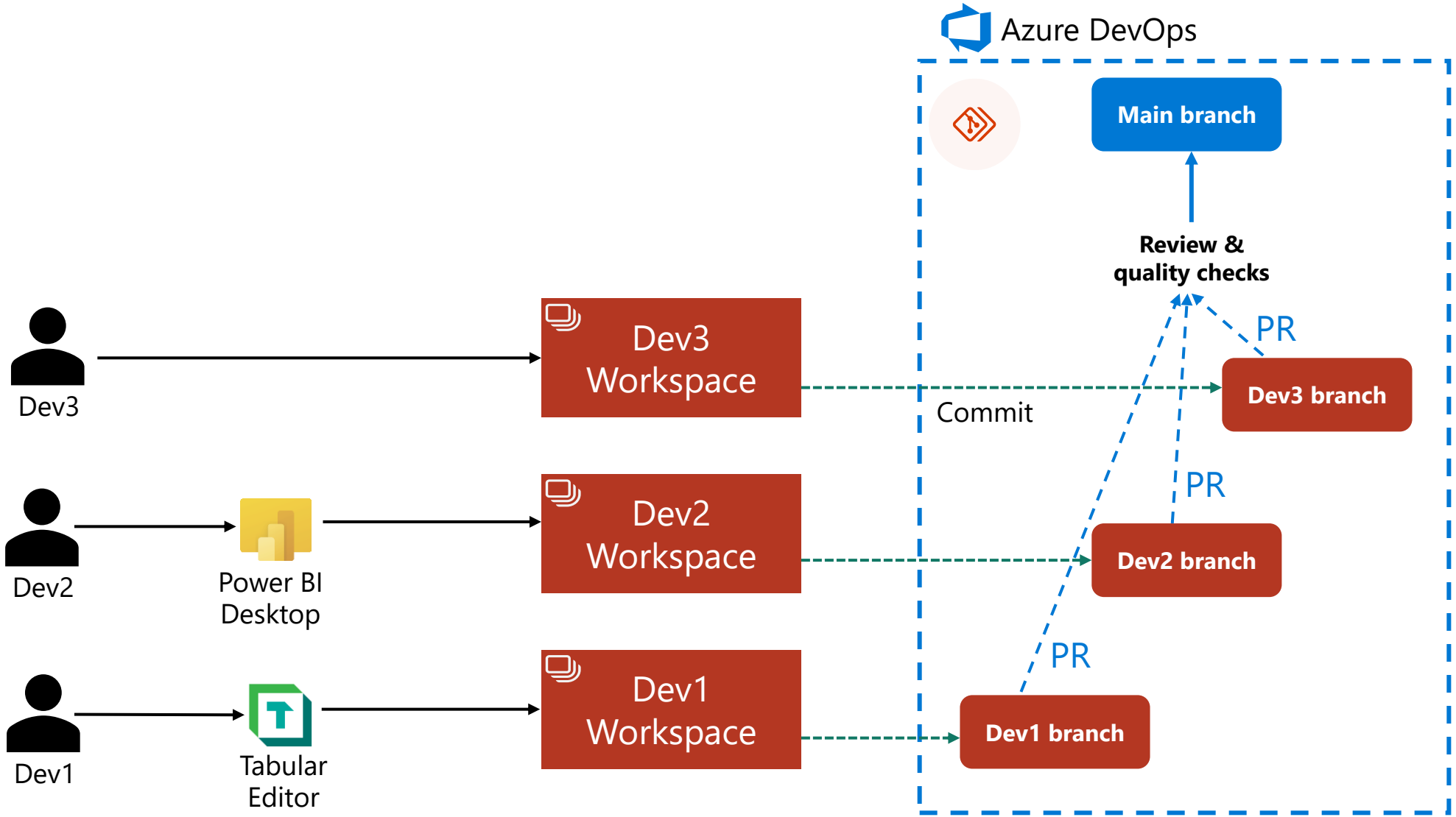
Development tools

Isolated development environments

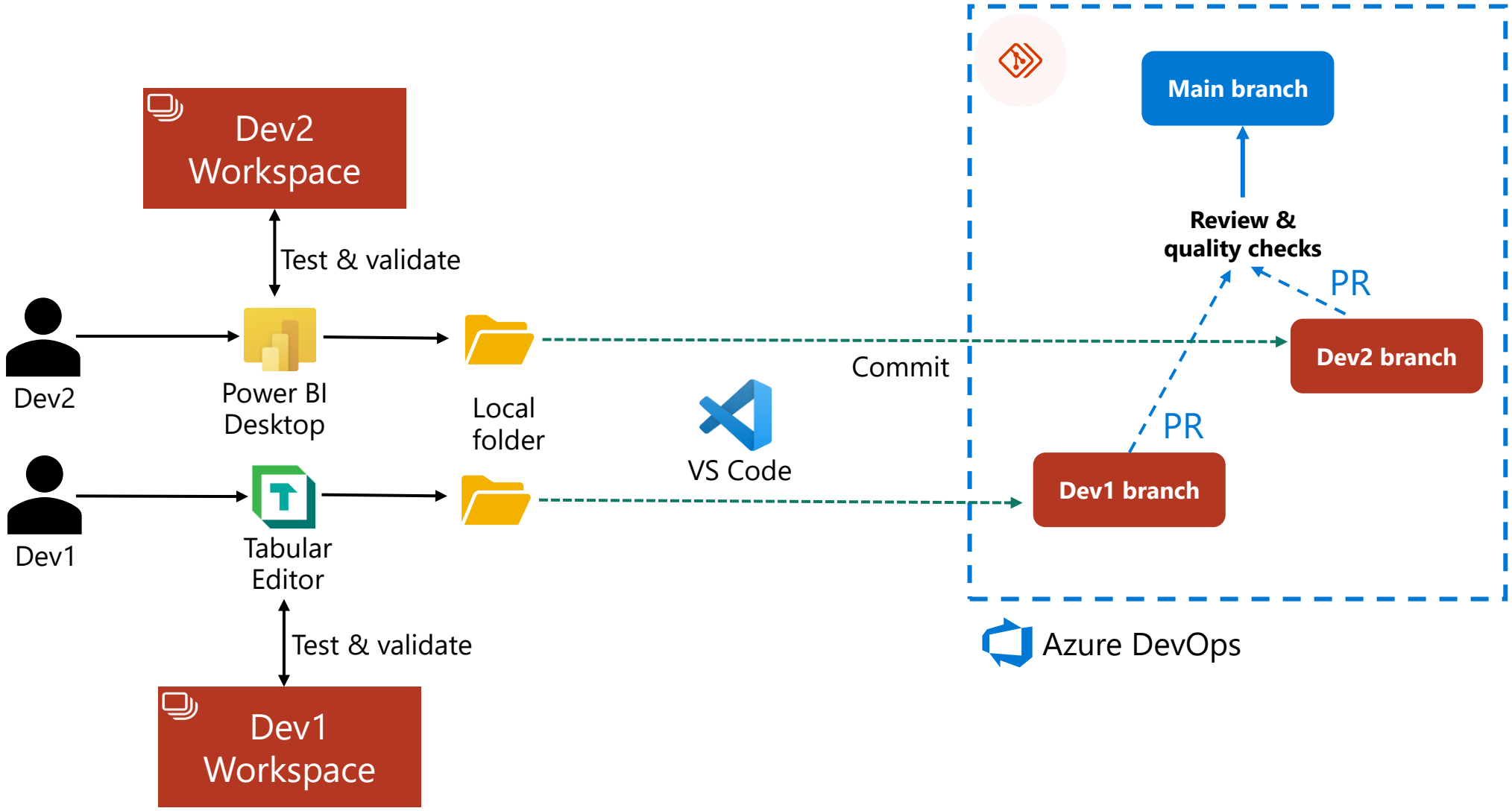
Review & quality checks

Can we use DevOps with PBIX files?

CI Strategy – Using Workspace



CI Strategy – Using Client Tools



CI Strategies – Comparison

	Using Workspace	Using Client Tools
Learning Curve	Low	High
Additional Tools	No	Yes (Git, VS Code with extensions)
Cherry picking	No	Yes
Merge conflict resolution	Limited	Flexible
Operational features	Limited • can't create pull requests • can't visualize change history	Extended
Supported Power BI file types	PBIX, PBIP (TMSL, TMDL)*	PBIP (TMSL, TMDL)* only
Supported file types	Fabric only	All

Continuous Deployment (CD)

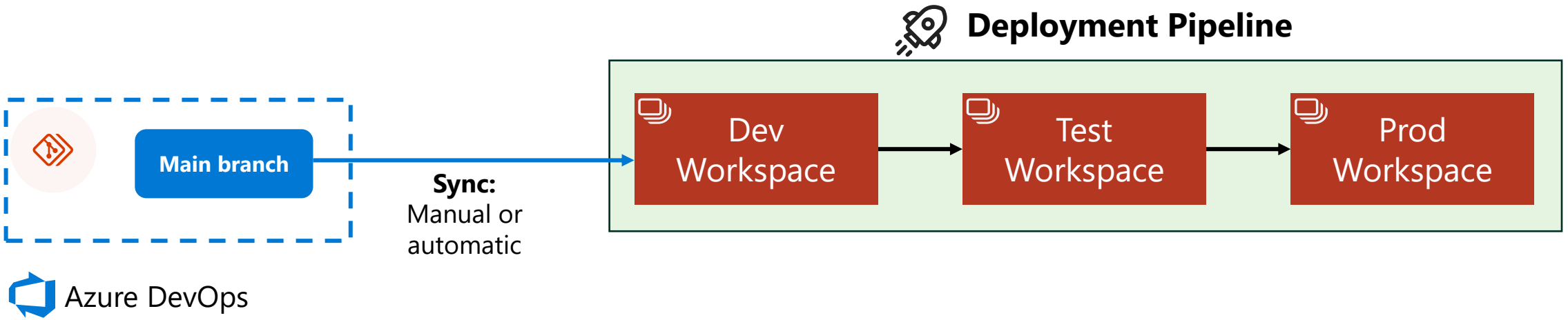
Defining a CD Strategy

Feature parity across environments

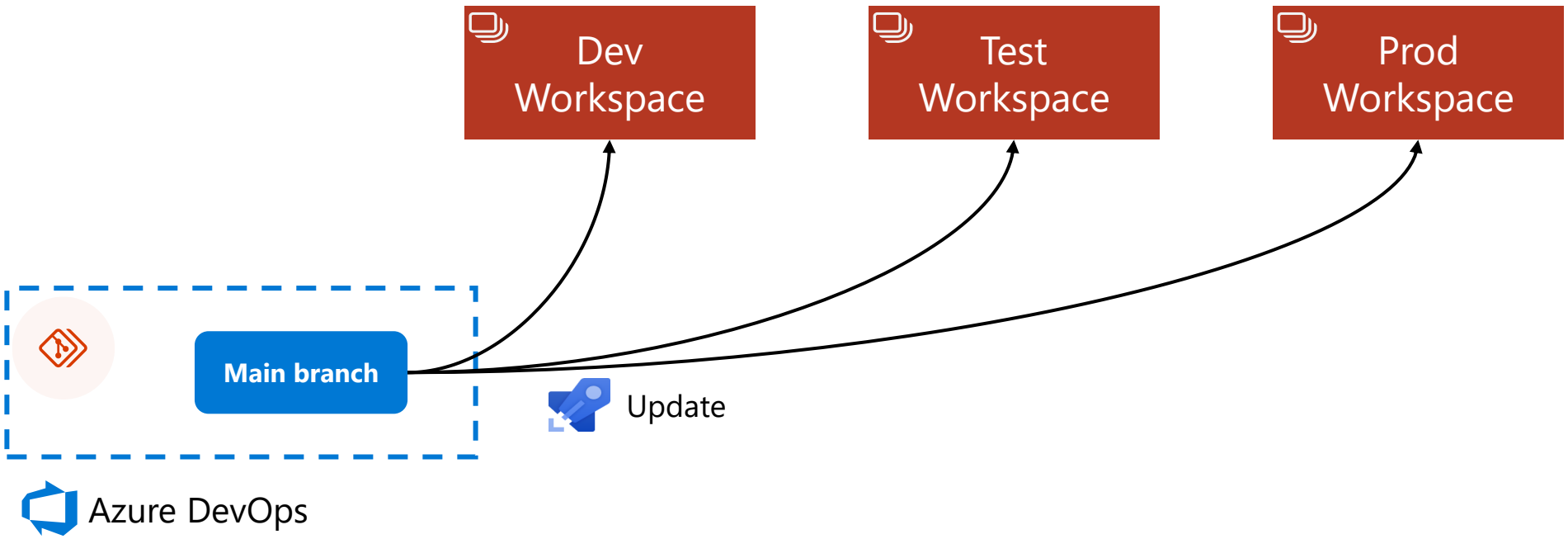
Custom changes during deployments

Complexity

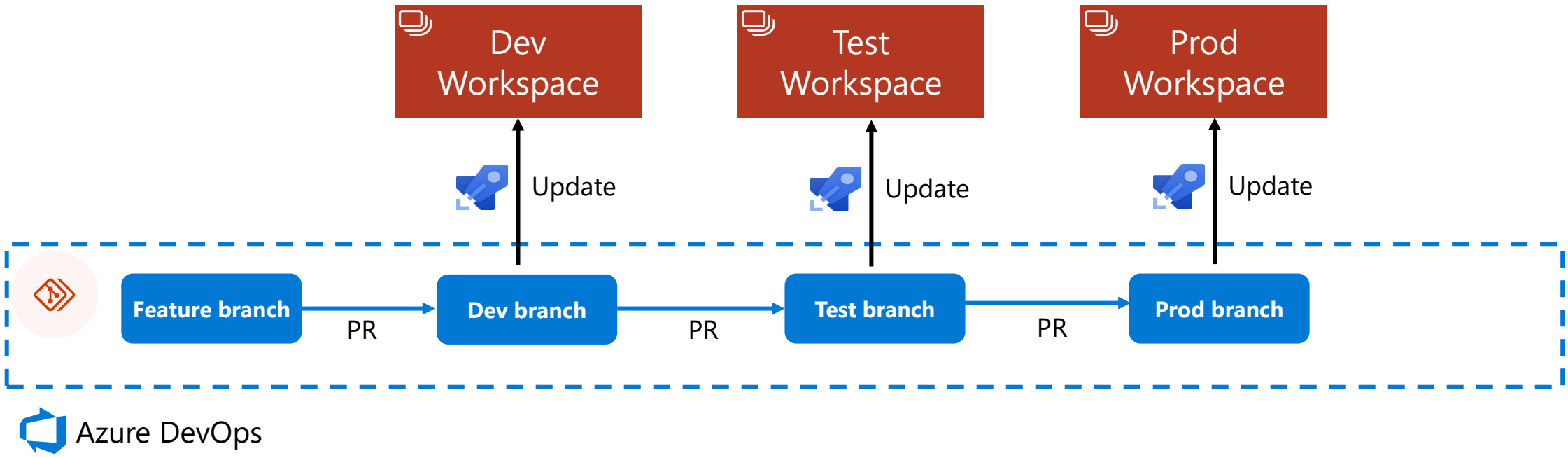
CD Strategy – Fabric Deployment Pipelines



CD Strategy – Git based Single Main Branch



CD Strategy – Git based Multiple Main Branches



CD Strategies – Comparison

	Fabric Deployment Pipelines	Git – Single Main Branches	Git – Multiple Main Branches
Learning Curve	Low	High	Highest
Setup	Simple	Complex	Complex
Complexity	Low	High	Highest
Maintenance	Low	High	Highest
Cherry picking features	No	No	Yes
Custom changes during deployments	No	Yes	Yes
Automation	Low	High	High

Defining a CI/CD strategy for your team

Development tools required

Training and learning curve

Setup and maintenance complexity

Power BI only or Fabric items

Flexibility and automation capabilities

Combine & evaluate various CI/CD strategies

Recommendations & Best Practices

- Connect dev workspace to Git and use deployment pipelines
- Use client tools for items that support it
- Developer should work in isolated environments
- Enforce reviews and automated quality checks
- Merging non-working code on the main branch should be punishable by DEATH
- Set up branch naming conventions
- Provide detailed commit messages

Takeaways

Train and promote DevOps in your teams

Automate quality checks

Combine and evaluate different CI/CD strategies

Thank You!!

Dhyanendra Singh Rathore

 dhyans