Application Lifecycle Management for Business Intelligence

A comprehensive framework for managing Power BI solutions from requirements through maintenance

Power BI & Fabric SUMMIT 2026

Download My Deck

Power BI & Fabric SUMMIT 2026

Special thanks to Fabric and Power BI Team at

Microsoft

This Summit presented to you by

RADACAD

Power BI & Fabric SUMMIT 2026

Michael McKinley

- Michael McKinley is a data analytics consultant, instructor, and Power BI evangelist with a passion for turning complexity into clarity. At White Cap Supply, he leads data intelligence efforts that empower senior leaders with actionable insights.

- Outside the office, Michael coleads the Nashville Power BI User Group, teaches live training courses, and speaks at conferences to help others harness the full potential of Power BI, the Power Platform, and Microsoft Fabric.

Executive Summary

Application Lifecycle Management (ALM) in Power BI provides a structured approach to developing, deploying, and maintaining Power BI solutions, ensuring consistency, quality, and scalability across quality, and scalability across the organization.

Structured Development

ALM brings structure and repeatability to Power BI development, reducing errors and and accelerating delivery.

Risk Reduction

Reduces risk through version control, testing protocols, and documented processes.

Complete Lifecycle Coverage

Covers the complete lifecycle: from gathering requirements through ongoing maintenance maintenance and enhancement.

Enhanced Collaboration

Improves collaboration between business stakeholders, developers, and support teams.

Enterprise Scaling

Essential for organizations scaling beyond ad-hoc report creation to enterprise-grade grade analytics.

Faster Time-to-Value

Enables faster time-to-value through standardized deployment and change management practices.

What We'll Cover

Section 1: Requirements & Planning Planning

Section 2: Documentation

Section 3: Version Control & Deployment

Section 4: Testing & Quality Assurance

Section 5: Release & Adoption

Section 6: Support & Maintenance

Section 7: Change Management

What We'll Cover

Section 1: Requirements & Planning

Section 2: Documentation

Section 3: Version Control & Deployment

Section 4: Testing & Quality Assurance

Section 5: Release & Adoption

Section 6: Support & Maintenance

Section 7: Change Management

Section 1: Requirements & Planning Overview

Identifying Key Stakeholders

Identify decision makers, end users, and data owners who will influence or be influence or be affected by the solution.

Defining Business Requirements

Document the business challenge, key questions to answer, decisions to be to be made, and success metrics.

Technical Requirements and Security

Establish data refresh schedules, performance expectations, security protocols, protocols, and access controls.

Scope Definition and Boundaries

Define project timeline, milestones, deliverables, and what's explicitly out of out of scope to prevent scope creep.

Section 1: Requirements & Planning Overview

Identifying Key Stakeholders

Identify decision makers, end users, and data owners who will influence or be influence or be affected by the solution.

Defining Business Requirements

Document the business challenge, key questions to answer, decisions to be to be made, and success metrics.

Technical Requirements and Security

Establish data refresh schedules, performance expectations, security protocols, protocols, and access controls.

Scope Definition and Boundaries

Define project timeline, milestones, deliverables, and what's explicitly out of out of scope to prevent scope creep.

Identifying Key Stakeholders

Successful Power BI projects begin with clearly identifying everyone who will influence or be affected by the solution. This foundational step ensures alignment and prevents costly rework later in the development cycle.

Decision Makers

Executive sponsors and business leaders who approve the project, set priorities, and allocate resources. They define success criteria and ensure strategic alignment.

End Users

The people who will interact with reports daily. Understanding their technical proficiency, workflow patterns, and pain points is critical for user adoption.

Data Owners

Subject matter experts who understand source source systems, data quality issues, and business rules. They validate data accuracy and accuracy and approve semantic models.

Document each stakeholder's role, expectations, and communication preferences early. This clarity prevents scope creep and ensures everyone understands their everyone understands their responsibilities throughout the project lifecycle.

Power BI & Fabric SUMMIT 2026

Defining Business Requirements

Problem Definition

What specific business challenge are we we solving? Document the current state, state, pain points, and desired future state. state. Quantify the impact where possible. possible.

Decisions to Be Made

What actions will users take based on the the insights? Understanding decision workflows ensures reports support actual actual business processes, not just passive passive consumption.

Key Questions

What questions must the solution answer? List the top 5-10 analytical questions that drive business value. Prioritize them based on impact.

Success Metrics

How will we measure if the solution succeeds? Define clear, measurable criteria criteria tied to business outcomes, not just just technical delivery.

Technical Requirements and Security

Technical requirements translate business needs into actionable specifications. This phase identifies constraints, dependencies, and technical feasibility before development begins.

1

Data Sources and Availability

Catalog all required data sources. Verify access permissions, API availability, and availability, and data quality. Identify gaps that need resolution before development.

2

Refresh Frequency Needs

How current must the data be? Real-time, hourly, daily, or weekly? Balance business needs against system capacity and costs.

3

Performance Expectations

Define acceptable query response times and report load speeds. Large datasets Large datasets or complex calculations may require optimization strategies from strategies from the start.

4

Security Requirements

Who can see what data? Document row-level security needs, sensitivity labels, sensitivity labels, and compliance requirements. Security must be designed in, designed in, not bolted on later.

Scope Definition and Boundaries

Clear scope boundaries prevent project creep and set realistic expectations. This critical step defines what will defines what will and won't be delivered, establishing a shared understanding with stakeholders.

01

In-Scope vs. Out-of-Scope

Create explicit lists of included and excluded features. features. Document deferred items in a backlog for for future consideration. Get stakeholder sign-off on off on these boundaries.

02

Phased Approach vs. All-at-Once

Will you deliver incrementally or in one release? Phased approaches reduce risk and enable faster feedback but require careful dependency management.

03

Timeline and Resource Constraints

How much time and effort is available? Identify team capacity, competing priorities, and hard deadlines. Adjust deadlines. Adjust scope to match available resources realistically.

Section 2: Documentation Overview

Architecture Documentation

Document data source mapping, workspace structure, and data flow diagrams to visualize the complete solution architecture.

Technical Specification

Capture DAX measures, data model relationships, calculated columns, and transformation logic for developer reference.

Refresh, Dependencies, and Security

Define refresh schedules, data dependencies, gateway configurations, and and security protocols including RLS & OLS implementation.

Support Documentation

Create user guides, FAQs, troubleshooting steps, and contact information to information to enable self-service support.

Section 2: Documentation Overview

Architecture Documentation

Document data source mapping, workspace structure, and data flow diagrams diagrams to visualize the complete solution architecture.

Technical Specification

Capture DAX measures, data model relationships, calculated columns, and columns, and transformation logic for developer reference.

Refresh, Dependencies, and Security

Define refresh schedules, data dependencies, gateway configurations, and and security protocols including RLS & OLS implementation.

Support Documentation

Create user guides, FAQs, troubleshooting steps, and contact information to information to enable self-service support.

Architecture Documentation

Data Source Mapping

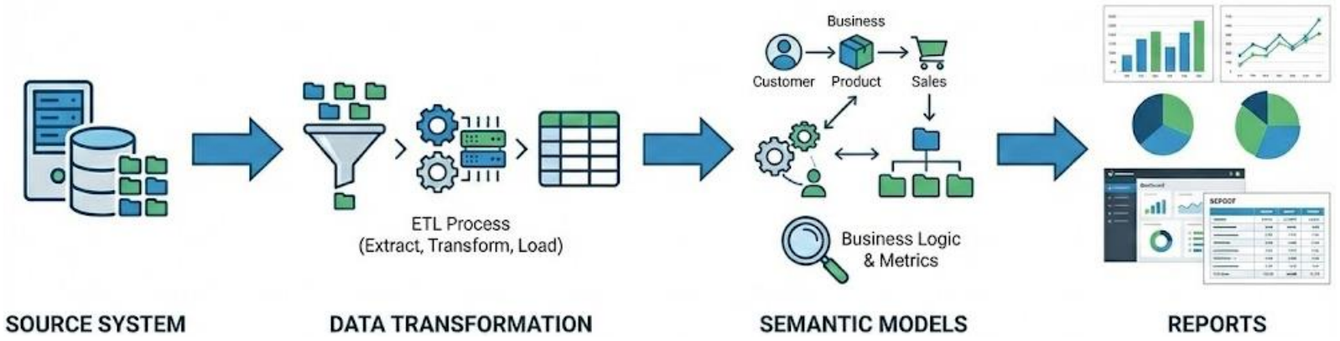
Document every data source with connection details, refresh credentials, and business owners. Include database names, table structures, and API endpoints.

Workspace Structure

Map out your workspace hierarchy and naming conventions. Show how Development, Test, and Production environments are organized within Fabric.

Data Flow Diagrams

Visualize the complete data journey: source systems → data transformation → semantic models → reports. Include refresh dependencies and gateway configurations.
configurations.



Technical Specification

Technical documentation enables developers to understand, maintain, and extend your solution. Comprehensive technical docs reduce debugging time and prevent knowledge silos.

Table, Column, & Measure Descriptions

Every table, column, and measure should have a clear description explaining its business purpose, data source, and any transformations applied. Use the Description property in Power BI for in-tool documentation.

Measure Definitions and Business Logic

Document DAX measures with comments explaining calculation logic, business rules, and any assumptions. Include examples of expected results and edge cases to handle.

Relationships and Data Model Design

Document your star schema design, relationship cardinality, and cross-filter direction choices. Explain why certain modeling decisions were made to guide future modifications.

Refresh, Dependencies, and Security

Refresh Schedules

Document when each dataset refreshes, including time zones and frequency. Map out frequency. Map out dependencies—e.g., Dataset B can't refresh until Dataset A Dataset A completes. Include SLA expectations.

Gateway Configuration

Record gateway locations, assigned administrators, and data source connections. connections. Document any firewall rules or network requirements for troubleshooting connectivity issues.

Row-Level and Object-Level Security

Clearly document RLS roles, filter expressions, and test scenarios. For OLS, document document which objects are restricted and to whom. Include validation steps to verify steps to verify security works correctly.

Support Documentation

Support documentation helps users find answers independently and guides support teams when issues arise. Comprehensive support docs reduce ticket volume and improve user satisfaction.

Where to Find Assets

Provide clear paths to reports, datasets, and workspaces. Include URLs, workspace names, and app locations. Create a central index that's easy to search.

Contact Information

List who to contact for different issues: data questions, technical problems, enhancement requests. Include response time expectations and escalation paths.

Known Issues and Limitations

Document current limitations, workarounds, and planned fixes. Being Being transparent about known issues reduces frustration and unnecessary unnecessary support tickets.

Useful Resources

Link to training materials, user guides, and FAQ documents. Include video tutorials for common tasks and best practices for using the solution effectively.

Section 3: Version Control Overview

Version Control Fundamentals

Understand why version control matters, available tools like Azure DevOps and GitHub, and what files to version in Power BI projects.

Option 1: In-Service Deployment Pipelines

Use Fabric's native deployment pipelines for simple multi-environment setup with Development, Test, and Production workspaces.

Option 2: External Git with Feature Branches

Implement feature branch workflow with external Git for granular change tracking, code review, and parallel development.

Option 3: Three Permanent Branches

Maintain separate Dev, Test, and Main branches that mirror your environments for structured promotion and release management.

Section 3: Version Control Overview

Version Control Fundamentals

Understand why version control matters, available tools like Azure DevOps and DevOps and GitHub, and what files to version in Power BI projects.

Option 1: In-Service Deployment Pipelines

Use Fabric's native deployment pipelines for simple multi-environment setup environment setup with Development, Test, and Production workspaces. workspaces.

Option 2: External Git with Feature Branches

Implement feature branch workflow with external Git for granular change change tracking, code review, and parallel development.

Option 3: Three Permanent Branches

Maintain separate Dev, Test, and Main branches that mirror your environments environments for structured promotion and release management.

Version Control Fundamentals

Understanding version control basics is essential for modern Power BI development. It provides the foundation for collaborative work, change tracking, and tracking, and deployment automation.

Why It Matters

Version control tracks every change, enables enables team collaboration, supports code code review, provides rollback capabilities, and capabilities, and creates an audit trail for compliance.

Available Tools

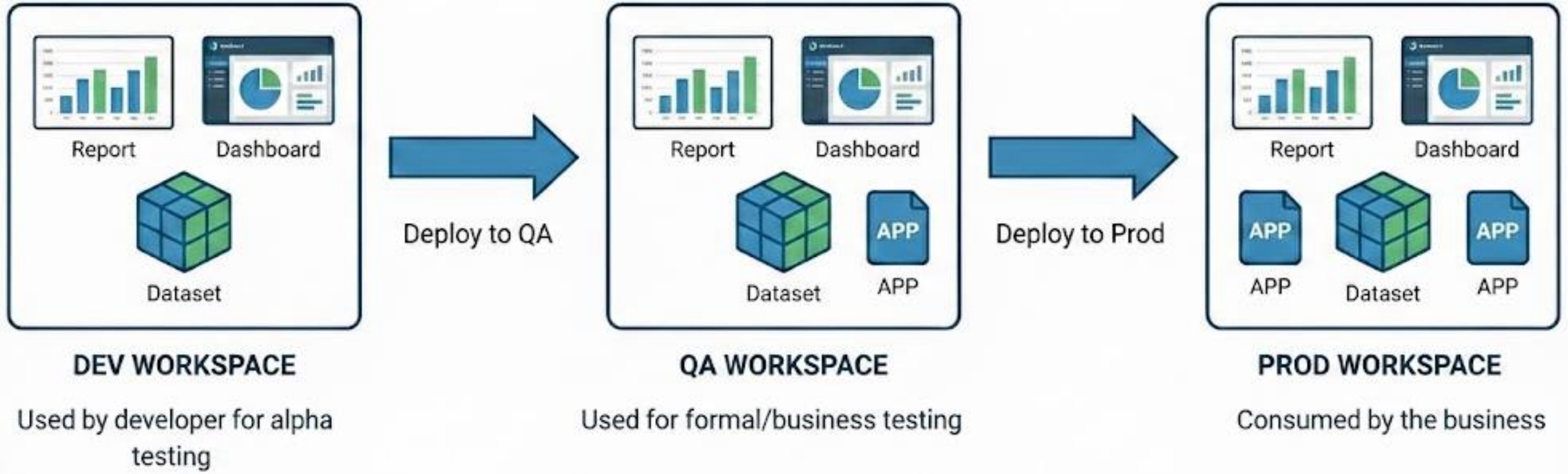
Azure DevOps and GitHub are the primary primary options for Power BI. Both integrate integrate with Fabric workspaces and support support PBIP files for granular version control. control.

What to Version

Version PBIP project files, dataset definitions, definitions, report layouts, DAX measures, measures, Power Query transformations, and and configuration files. Exclude sensitive credentials and data caches.

Branching Strategy Option 1: In-Service Pipelines

Deployment pipelines within Fabric provide the simplest path to multi-environment development. This approach leverages Power BI's native capabilities without requiring external Git integration.



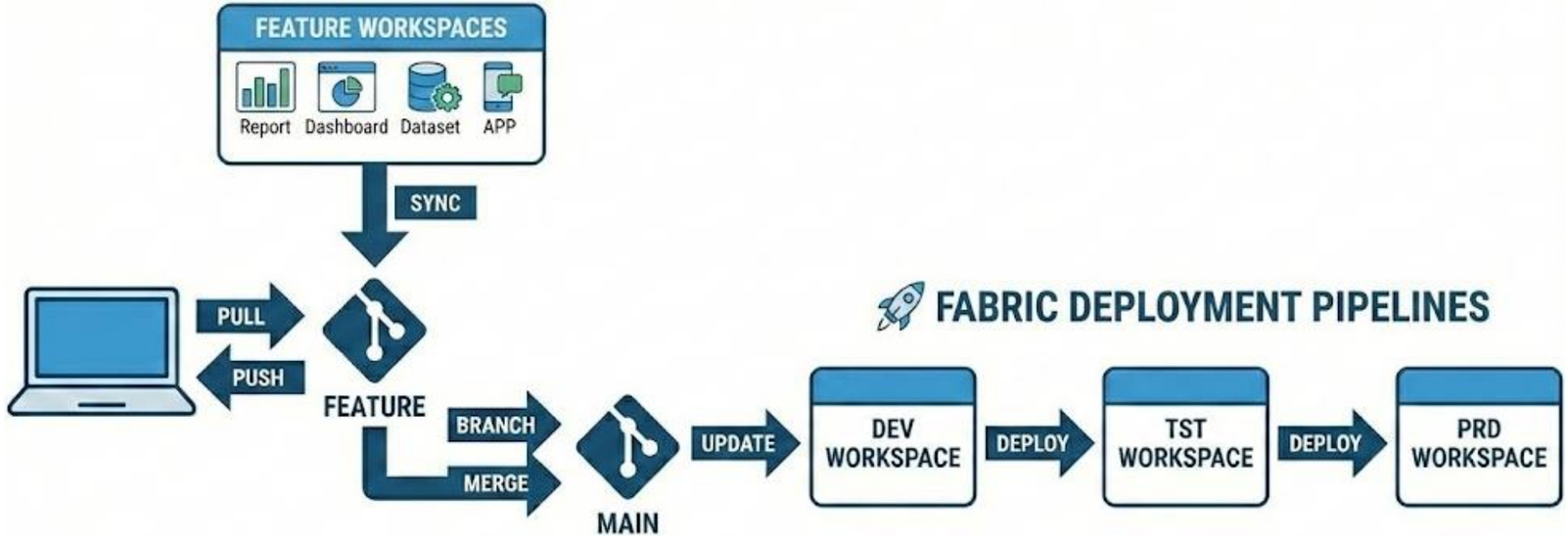
Best for: Teams new to version control or those wanting simplicity over granular change tracking. Provides basic promotion workflow without Git complexity.

Branching Strategy Option 2: External Git with Feature Branches

The Workflow

Developers create feature branches for each enhancement or bug fix. Changes are tested in isolation, peer-reviewed via pull requests, then merged into main. The main branch syncs to the Development workspace, and in-Service deployment pipelines promote to Test and Production.

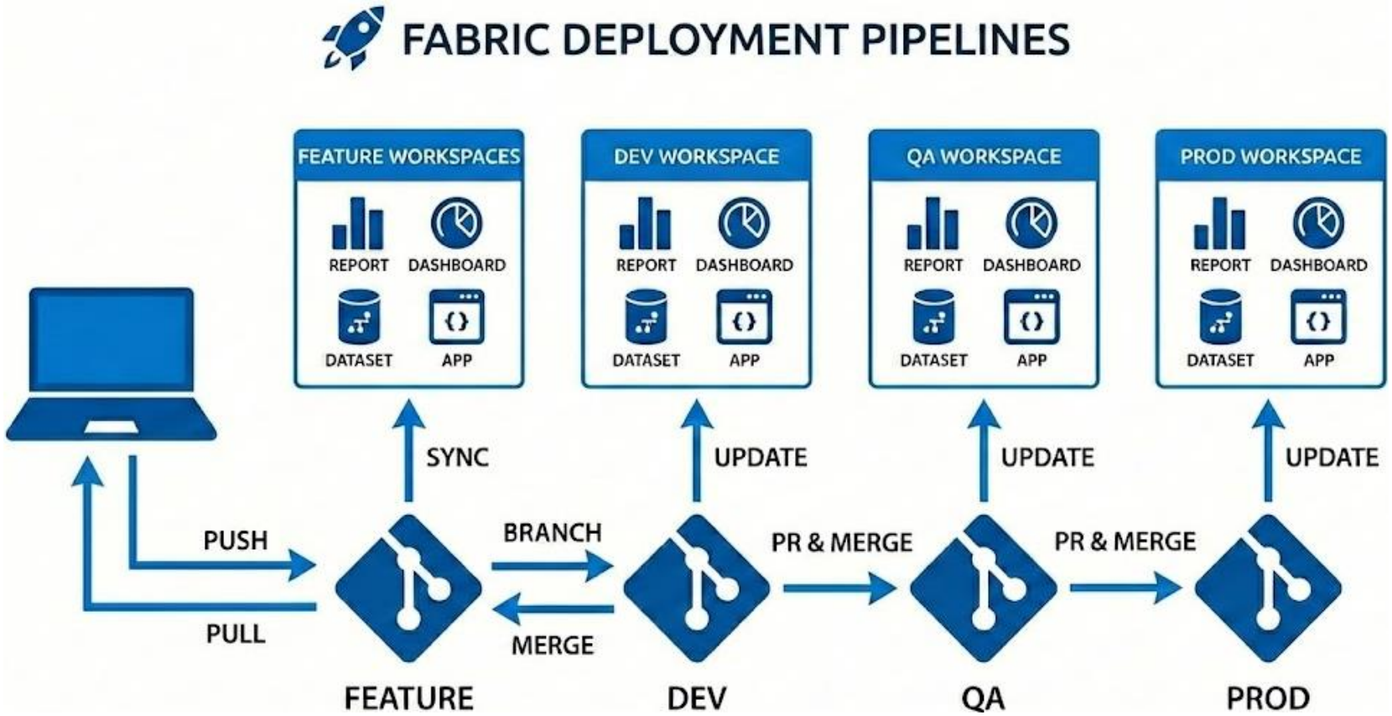
This strategy provides maximum flexibility and traceability, enabling multiple developers to work simultaneously without conflicts.



Best for: Mature teams with strong Git skills who need parallel development, code review processes, and granular change tracking. Requires more setup but provides enterprise-grade version control.

Branching Strategy Option 3: Three Permanent Branches

This strategy maintains separate permanent branches for Development, Test, and Production environments. Changes flow between branches through manual merges or automated merges or automated CI/CD pipelines.



Best for: Teams wanting clear environment separation with explicit promotion between stages.

Section 4: Testing & Quality Assurance Overview

Testing Strategies and Code Quality

Establish comprehensive testing approaches including data validation, DAX testing, visual testing, and user acceptance testing with clear quality standards.

Automated Validation Approaches

Implement automated tools and scripts to scale QA scale QA efforts, catch issues early, and maintain maintain quality consistently across development development cycles.

Approval Gates and Sign-Off

Define clear approval responsibilities and sign-off processes based on change type to ensure proper review before production deployment.

Section 4: Testing & Quality Assurance Overview

Testing Strategies and Code Quality

Establish comprehensive testing approaches including data validation, DAX testing, visual testing, testing, and user acceptance testing with clear clear quality standards.

Automated Validation Approaches

Implement automated tools and scripts to scale QA scale QA efforts, catch issues early, and maintain maintain quality consistently across development development cycles.

Approval Gates and Sign-Off

Define clear approval responsibilities and sign-off sign-off processes based on change type to ensure ensure proper review before production deployment.

Testing Strategies and Code Quality

Comprehensive testing catches issues early and maintains solution quality as complexity grows. Establish clear standards that balance thoroughness with development velocity.
velocity.

Data Validation

Verify data accuracy by comparing Power BI outputs against source systems. Test edge
Test edge cases, null values, and aggregation logic. Validate row counts, totals, and key
and key metrics.

DAX Testing

Test measures with known inputs and expected outputs. Verify calculations across
across different filter contexts. Check for proper handling of blank values and division by
division by zero scenarios.

Visual Testing

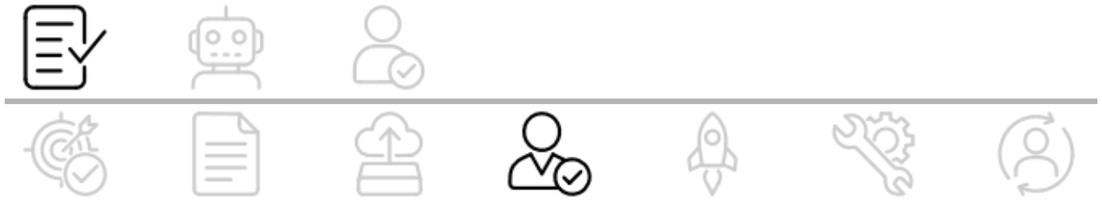
Confirm visuals display correctly across different screen sizes and browsers. Test
interactivity: slicers, drill-through, tooltips. Verify performance with realistic data volumes.

User Acceptance Testing

Have actual users test in Test environment. Validate reports answer their business
questions. Gather feedback on usability and performance before production deployment.

Code Quality Standards

- Enforce consistent naming conventions for measures, tables, and columns
- Set optimization rules: disable auto date/time, remove unused columns, use variables in complex DAX
- Establish performance expectations



Automated Validation Approaches

Automation scales your QA efforts and catches issues that manual testing might miss. These tools integrate into your development workflow to maintain quality consistently.

Tabular Editor Best Practice Analyzer

Run BPA rules to check model structure, naming conventions, DAX optimization, model hygiene, metadata quality, etc. Create custom rules for your organization's standards. Integrate into CI/CD pipelines for automatic validation.

Power Query Linting

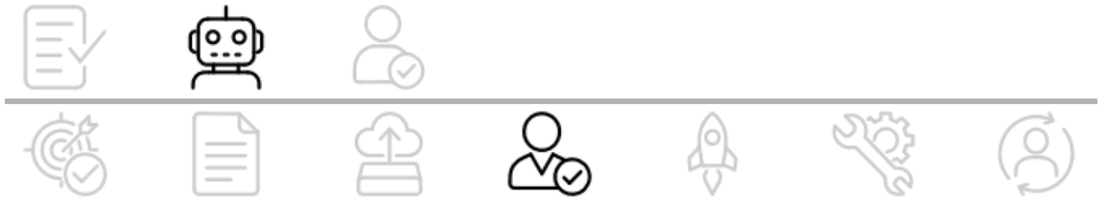
Use PQ Lint to validate M code formatting, detect inefficient queries, and enforce coding standards. Catches common mistakes like repeated folding breaks or unnecessary data type conversions.

DAX Query View Validation

Write DAX queries that assert expected measure results. Run these as regression tests after each change. Example: verify YTD Sales for known period equals expected value. Store queries in version control.

Performance Benchmarks with Azure DevOps

Use DAX Studio and Power BI REST APIs to measure query performance. Track metrics over time in Azure DevOps. Set thresholds and fail builds if performance degrades significantly.



Approval Gates & Sign-Off

Who Approves Changes

Define clear approval responsibilities based on change type:

- **Data model changes:** Data architect and data owner
- **New reports/features:** Business stakeholder and UX lead
- **Performance optimizations:** Technical lead
- **Security changes:** Security officer and business owner

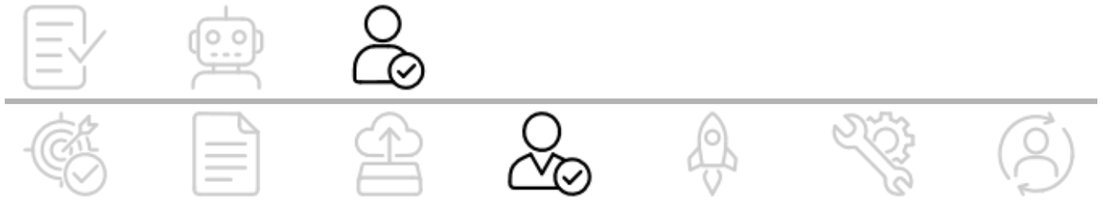
Avoid bottlenecks by empowering multiple approvers and setting approval approval SLAs. Use Azure DevOps or ServiceNow for formal approval approval workflows.

Sign-Off Process

Require formal sign-off before production deployment. Include:

- UAT completion confirmation
- Performance test results
- Security review checklist
- Rollback plan verification
- Documentation updates confirmation

Maintain sign-off records for audit trails. Digital signatures or email confirmations satisfy most compliance requirements.



Section 5: Release & Adoption Overview

Communication and Training

Build user adoption through pre-launch announcements, role-specific training sessions, office hours, and comprehensive onboarding materials that drive engagement.

Rollout Strategy and Success Tracking

Choose between phased or big bang deployment approaches, track adoption metrics, and provide intensive launch support to ensure successful user adoption.

Section 5: Release & Adoption Overview

Communication and Training

Build user adoption through pre-launch announcements, role-specific training specific training sessions, office hours, and comprehensive onboarding materials onboarding materials that drive engagement.

Rollout Strategy and Success Tracking

Choose between phased or big bang deployment approaches, track adoption adoption metrics, and provide intensive launch support to ensure successful successful user adoption.

Communication and Training

Effective communication transforms skeptical users into enthusiastic advocates. Start early, communicate often, and tailor messages to different audience different audience segments.

01

Pre-Launch Announcements

Announce the solution 2-3 weeks before launch. Explain the business problem being solved, benefits to users, and what's changing. Build excitement and set expectations. Use multiple channels: email, team meetings, intranet posts.

02

Training Sessions

Offer role-specific training: executives need high-level overviews, analysts need deep technical training. Provide live sessions, recorded videos, and hands-on workshops. Aim for 80% of users trained before launch.

03

Office Hours and Support

Schedule regular office hours where users can ask questions and get help. Staff help. Staff these with knowledgeable team members who can handle technical technical and business questions. Continue for an appropriate period of time of time post-launch.

04

Onboarding Materials

Create quick-start guides, video tutorials, and FAQ documents. Make them easily them easily discoverable—embed links in the reports themselves. Include real Include real business scenarios users can relate to.

Rollout Strategy and Success Tracking

Phased vs. Big Bang

Phased rollout: Deploy to one department or user group first. Gather feedback, make adjustments, then expand. Lower risk but slower time-to-value. Best for complex or politically sensitive solutions.

Big bang: Deploy to all users simultaneously. Faster adoption, consistent consistent experience. Higher risk of widespread issues. Best when when solution is well-tested and relatively simple.

Adoption Tracking

Monitor these metrics to gauge adoption:

- Unique users per week
- Report views and usage patterns
- Support ticket volume and types
- Time spent in reports (engagement)
- Feature utilization (filters, drill-through)

Set adoption targets: aim for 70% of target users active within 30 days. Identify power users who can champion the solution.

Launch Support and Feedback

Provide intensive support during the first two weeks. Collect user feedback through surveys, support tickets, and direct conversations. Act quickly on early feedback—quick early feedback—quick wins build momentum. Define success criteria upfront: adoption rates, business outcomes, user satisfaction scores.

Section 6: Support & Maintenance Overview

Support Structure and Access Management

Establish tiered support model from self-service to technical support, with clear escalation paths and access management processes for efficient issue resolution.

Monitoring and Observability

Implement proactive monitoring for usage analytics, performance metrics, and error detection to catch issues before users report them.

Ongoing Maintenance Activities

Maintain solution health through dataset refresh monitoring, performance optimization, security updates, and regular capacity management.

Business Continuity Planning

Protect against disruptions with documentation, knowledge transfer, backup strategies, and disaster recovery procedures to ensure operational resilience.

Section 6: Support & Maintenance Overview

Support Structure and Access Management

Establish tiered support model from self-service to technical support, with clear with clear escalation paths and access management processes for efficient issue efficient issue resolution.

Monitoring and Observability

Implement proactive monitoring for usage analytics, performance metrics, and metrics, and error detection to catch issues before users report them.

Ongoing Maintenance Activities

Maintain solution health through dataset refresh monitoring, performance performance optimization, security updates, and regular capacity management. management.

Business Continuity Planning

Protect against disruptions with documentation, knowledge transfer, backup backup strategies, and disaster recovery procedures to ensure operational operational resilience.

Support Structure and Access Management

A tiered support model efficiently handles user issues while protecting your team's time. Clear roles and escalation paths ensure problems get resolved quickly at the quickly at the appropriate level.

Tier 1: Self-Service

Users find answers in documentation, FAQs, and and training materials. Aim to resolve 40-50% of 50% of issues here through comprehensive, searchable knowledge base.

Tier 2: Business Support

Help desk or business analysts handle user questions about report interpretation, data definitions, and basic troubleshooting. Handle 30-Handle 30-40% of issues. SLA: respond within 4 within 4 business hours.

Tier 3: Technical Support

BI developers and architects handle complex complex technical issues: performance problems, problems, data model changes, integration issues. issues. Handle remaining 10-20%. SLA: respond respond within 8 business hours, critical issues issues within 2 hours.

Access Management

Establish clear processes for granting, modifying, and revoking access. Use security groups for role-based access rather than individual user assignments. Require manager approval for workspace Contributor access. Review access quarterly to remove unnecessary permissions. Document access request procedures and typical turnaround times.

Monitoring and Observability

Proactive monitoring catches issues before users report them. Comprehensive observability provides insights into usage patterns, performance trends, and system health.

Usage Analytics

Track active users, popular reports, and usage trends over time. Identify underutilized assets that might be candidates for retirement. Spot adoption drops that signal problems or training needs.

Performance Monitoring

Monitor query durations, visual render times, and refresh completion times. Set up alerts when metrics exceed thresholds. Use DAX Studio and Performance Analyzer to diagnose slow reports.

Error Detection and Alerting

Configure alerts for refresh failures, gateway connectivity issues, and capacity throttling. Use Power Automate or Azure Logic Apps to send notifications to support teams. Maintain error logs for troubleshooting and trend analysis.

Build monitoring dashboards that track key operational metrics. Share with stakeholders monthly to demonstrate solution health and proactive management. proactive management.

Ongoing Maintenance Activities

Dataset Refresh Monitoring

Check refresh logs daily. Investigate any failures immediately—users expect current data. Document common failure scenarios and resolutions. Consider implementing incremental refresh for large datasets to improve reliability and reduce refresh times.

Optimization

Regularly review and optimize slow-performing reports. Remove unused unused visuals and columns. Optimize DAX measures using variables and variables and query plan analysis. Archive old data that's no longer accessed. accessed. Schedule optimization sprints quarterly.

Capacity Management

Monitor capacity utilization trends. Watch for approaching limits on storage, memory, or CPU. Plan capacity upgrades before hitting constraints, not after users experience degraded performance. Optimize heavily-used datasets to reduce capacity pressure.

Security Patches and Updates

Stay current with Power BI monthly updates. Test major releases in Development before rolling to Production. Review Microsoft security advisories and apply patches promptly. Update gateway software and connection drivers regularly to prevent compatibility issues.

Business Continuity Planning

Business continuity planning protects your organization from disruptions caused by staff turnover, system failures, or disasters. Don't wait for a crisis to discover you lack recovery capabilities.

- **Key Person Risk Mitigation**

  Avoid single points of failure where only one person knows critical systems. Cross-train team members on essential tasks. Document tribal knowledge. Ensure at least two people can handle dataset refreshes, gateway management, and critical report maintenance.

- **Backup Strategy**

  Regularly export PBIX files and workspace content using Power BI REST APIs. Store backups in separate location from source. Version control provides automatic backups of dataset definitions. Maintain configuration backups: workspace settings, security groups, deployment pipeline configurations.

- **Disaster Recovery Procedures**

  Document step-by-step recovery procedures for workspace deletion, dataset corruption, or capacity outage scenarios. Test recovery procedures annually—simulated disasters reveal gaps in your plans. Define recovery time objectives (RTO) and recovery point objectives (RPO) for critical solutions.

- **Service Level Agreements**

  Establish clear SLAs for report availability, data freshness, and support response times. Document what users can expect during planned maintenance and unplanned outages. Communicate SLAs to stakeholders and track compliance monthly.

Section 7: Change Management Overview

Enhancement Request Process

Establish structured intake process for capturing, tracking, and prioritizing enhancement requests with clear submission workflows and prioritization frameworks.

Approval Workflows and Communication

Define decision makers and approval responsibilities based on change type, with clear business case requirements and stakeholder communication protocols.

Continuous Improvement

Create feedback loops through post-deployment surveys, usage analytics, and regular retrospectives to refine solutions and processes over time.

Section 7: Change Management Overview

Enhancement Request Process

Establish structured intake process for capturing, capturing, tracking, and prioritizing enhancement enhancement requests with clear submission workflows and prioritization frameworks.

Approval Workflows and Communication

Define decision makers and approval responsibilities responsibilities based on change type, with clear clear business case requirements and stakeholder stakeholder communication protocols.

Continuous Improvement

Create feedback loops through post-deployment deployment surveys, usage analytics, and regular regular retrospectives to refine solutions and processes over time.

Enhancement Request Process

A structured intake process captures enhancement ideas, evaluates their value, and manages stakeholder expectations. Good processes balance responsiveness with sustainable development sustainable development capacity.

Submission Process

Provide a simple way for users to submit requests: web web form, email, or ticketing system. Capture essential essential information:

- Business problem or opportunity
- Proposed solution
- Affected users and priority
- Success criteria

Intake and Tracking

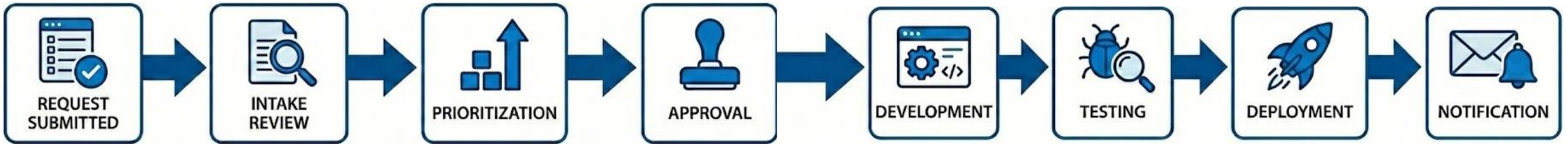
Log all requests in a tracking system (Azure DevOps, Jira, Jira, ServiceNow). Assign request IDs and send confirmation confirmation to requestor. Triage requests weekly—categorize by type, effort, and urgency. Track status from from submission through delivery.

Prioritization Framework

Score requests on business value, urgency, and effort. Use a simple matrix:

- High value, low effort: do first
- High value, high effort: plan carefully
- Low value, low effort: quick wins
- Low value, high effort: defer or decline

Publish prioritized backlog to manage expectations.



Approval Workflows and Communication

Decision Makers and Business Cases

Define who approves different request types. Small changes (cosmetic fixes, (cosmetic fixes, minor updates) may need only technical lead approval. Major approval. Major enhancements (new reports, data source additions) require require business stakeholder and IT leadership sign-off.

For significant requests, require business cases documenting expected ROI, resource requirements, and alignment with strategic goals. Use templates to standardize business case submissions.

Impact Assessment

Evaluate technical impact before approval: Will it affect existing reports? Does it require new data sources or security changes? What's the testing effort? Are there dependencies or risks? Document findings and share with decision makers.

Release Communication

Announce changes before deployment. Include what's new, what's changing, and what users need to know. Use multiple channels based on change scope:

- Minor updates: in-app notifications
- Feature additions: email announcements
- Major changes: presentations and training

Training and Documentation

Update training materials and documentation with each release. Offer refresher training for significant changes. Record short video walkthroughs for new features. Make it easy for users to find "what's new" information.

Continuous Improvement

Establishing robust feedback loops after deployment is crucial for refining your Power BI solutions, ensuring they meet evolving business needs, and fostering a needs, and fostering a culture of continuous improvement.

Post-Deployment Review

Conduct immediate reviews after each deployment to assess performance, identify initial bugs, and gather user reactions. This helps validate the change and catch critical issues early.

Continuous Improvement

Use insights from reviews to drive iterative enhancements. Implement a backlog of small, frequent improvements based on user feedback and performance metrics, rather than waiting for large-scale changes.

Lessons Learned

Document findings from reviews and improvement cycles to inform future projects. projects. Share successes and challenges, update update best practices, and refine development development processes to build institutional institutional knowledge.

Key Takeaways and Next Steps

ALM is a Journey, Not a Destination

Start with fundamentals—requirements and documentation—then progressively adopt version control, automated testing, and formal change management as your team matures.

Documentation Enables Everything Else

Without good documentation, version control and maintenance become exponentially harder. Treat documentation as a first-class deliverable, not an afterthought.

Balance Process with Agility

Don't let process become bureaucracy. Tailor ALM practices to your team size and solution complexity. Small teams need lighter processes than enterprise deployments.

Adoption Requires Ongoing Investment

Successful Power BI solutions need continuous support, training, and enhancement. Budget for 20-30% of development effort for ongoing maintenance ongoing maintenance and user support.

Measure What Matters

Track metrics that demonstrate business value: user adoption, decision speed, time saved, revenue impact. Technical metrics matter, but business outcomes justify continued investment.

Thank you

- Connect with me at:
  - /in/michaellmckinley
  - @_mikemckinley_
  - https://www.mckinley.consulting

LinkedIn

Deck

- Stay online for my live Q&A sessions