

# Top Impact Opportunities to Improve the Professional Developer Experience

Bruce Phillips, Matthew Roche

November 2025

## **CAT Research:**

Amy Sousa, Avishag Spillinger, Dawn Ferguson,  
Frank Huang, Paula Bach, Samira Jain, Troy  
Weekes, Veronika Monohan

## **CAT Programs:**

Dominic Cerchio, Fernando Lopez,  
Keerthi Thomas, Vanessa Araujo,

# Professional Developer Program

- **Why:** There is a disconnect between measured NPS and anecdotal feedback leadership receives
- **What we've done:** CAT Programs team has created a program to collect on-going feedback from professional developers
- **What we are showing you today:** Prioritized high-impact opportunities to improve the professional developer experience
- **Action Item:** Identify the opportunities relevant to you; use the program to engage with professional developers to learn more

# Who is a professional developer on Fabric?

A Professional Developer on Fabric:

- Builds and operates production-grade, governed, and automated data solutions on Microsoft Fabric as a core part of their role.
- Works across multiple workloads, including Data Engineering and/or Data Integration.
- Applies software engineering rigor (Git, CI/CD, testing, governance, security)
- Works on solutions for critical business processes that support downstream users.
- Is beyond the 85<sup>th</sup> percentile in regular Fabric usage



[Pro Dev Connect](#) – CAT Program information

[Racing Cars and Armored Trucks](#) – some thoughts on types of pro-developers

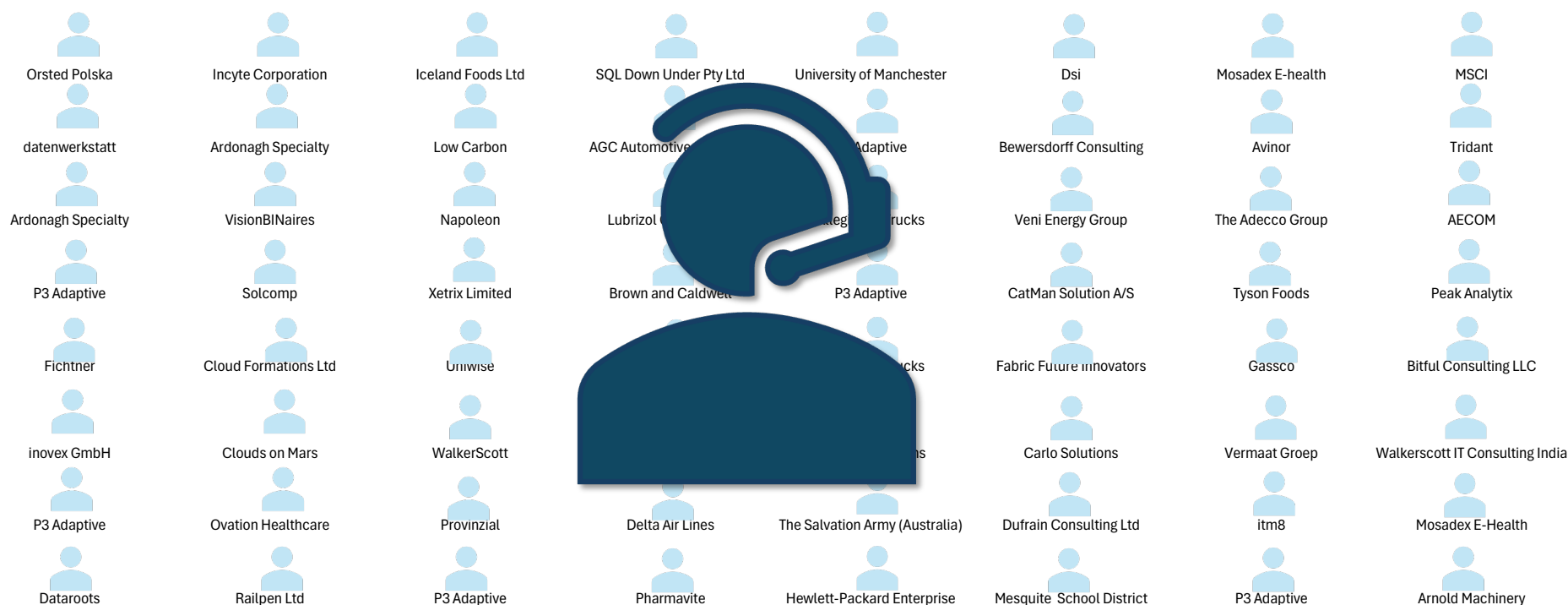
[Fabric Professional Developer Program.docx](#) – specific pro-dev recruiting criteria

# Research objectives

- Identify and prioritize **the most critical issues** faced by professional developers using Fabric
- Provide **clear recommendations** to guide investments for **maximum impact** on professional developer productivity and satisfaction

# We interviewed many, many professional developers

CAT Pro-Dev Connect Program - [56 hours of 1:1 Interviews](#)



Interviewers: Amy Sousa, Frank Huang, Avishag Spillinger, Paula Bach, Samira JaVeronika Monohan, Matthew Roche, Troy Weekes

# We extracted over 200 issues from these interviews

Shortcut Management: Permissions and ownership issues with shortcuts, especially when relying on OAuth, causing failures when developers leave. [14:12:06]	Limited customization options for organizational branding and domain identification in apps and workspaces, causing user confusion and extra support needs. 7	Difficulty with fine-grained access control in Fabric, requiring the team to build a complex inventory and auditing solution to monitor activities and enforce policies. 31:31	Checkpoint Creation Issues: User encountered problems with checkpoints not being created properly in lakehouse tables, resulting in errors and difficulties viewing files. He had to develop a notebook to restore checkpoints and optimize tables, which ultimately improved performance but highlighted gaps in Fabric's reliability. 19:52	Monitoring and Reporting: Built-in monitoring is seen as limited, with a preference for open-source solutions like FOAM reporting. The user wants more granular, global, and cost-effective monitoring options. 46:57	Starter Spark pools are oversized and not cost-effective for development, while custom pools have long startup times, causing delays and inefficiency 24:51
Manual, fragmented access control for SQL endpoint permissions, not integrated into Fabric CICC, causing governance and security concerns. 35:25	Governance and Security Complexity: Dan discussed the challenge of implementing granular security and governance, especially when integrating with external systems like Databricks. He emphasized the difficulty of providing appropriate access without overexposing sensitive data or requiring excessive permissions. 13:56	Security concerns with the Spark engine, as it allows open connections to any URL, posing a data exfiltration risk until recent improvements. 28:02	CICD and ALM Gaps: User pointed out major gaps in Fabric and Power BI regarding CICC, especially since his organization cannot use GitHub and must rely on GitLab, which complicates deployment pipelines and versioning. 49:06	Scalability and UAT Concerns: He expressed concerns about future scalability with KQL, as massive user acceptance testing has not been possible yet, and optimization is required from the beginning to avoid compute bottlenecks. 56:21	Governance and Cost Concerns: Nagesh raised concerns about using Purview for governance within Fabric, noting that it is an expensive proposition given their multi-tenant model and large client base. They are evaluating how much of Fabric's governance framework is practical for their needs. 50:19
Security limitations in Fabric lakehouse tables require the use of views for row-level security, as direct table security is unavailable. 44:42	Governance and Security Limitations: User mentioned that table-level security in KQL is not effective yet, requiring workarounds such as filtering queries and creating separate dashboards for different security groups. 54:37	The unified catalog in Purview is poorly integrated with Fabric and is seen as too bureaucratic, leading to a shift toward using the OneLake catalog despite its own limitations. 40:11	Slowly Changing Dimensions (SCD) Handling: He described difficulties implementing SCD Type 2 in Fabric, especially with large incremental datasets and updating records with historical changes, noting that the lakehouse struggles with small files and merges, while the warehouse is better but not ideal for heavy transactional workloads. 41:01	Scale and Performance Constraints: Louis described hitting data volume limits in Power BI (around 10 million rows was manageable, but scaling to 100 million rows for multiple years became too slow), which led him to move to Fabric for better performance and orchestration. 7:45	Cost Concerns: Lutz highlighted that cost is the main factor in evaluating Fabric features, mentioning shock at the high costs of Event House and Dataflow Gen2 compared to previous solutions. 13:55
Security and governance features are inconsistent, with some controls only available at code level and others in the UI, leading to confusion and extra manual steps. 32:25	Manual Steps for Preview Features: Robin mentioned that enabling data access roles in lake houses requires manual activation of a preview feature, which cannot be automated via API, resulting in unnecessary manual work for each data product. 32:56	No support for private package repositories, which is a security requirement for large organizations, and currently only public sources like PyPI can be used 44:04	CICD and Deployment Automation: The interviewee highlighted major pain points with automating deployments in Fabric, noting the lack of a dedicated CICC engineer and difficulties in automating processes, especially for Gen2 connectors. Manual redeployment and fragile processes were required due to missing deployment pipeline variables. 13:08	Capacity and Reliability Concerns: Gen2 connectors were described as heavy on capacity and less reliable, leading to operational challenges and a push to transition to pipelines and notebooks for better stability. 50:22	Feature Evaluation Process: Lutz described a process for testing Fabric features in a dedicated capacity, monitoring usage and costs, and gathering feedback from users before wider adoption. 11:51
Ownership and Scheduling Issues: Artifacts are owned by the creator's identity, causing failures when consultants leave or accounts are deactivated. Scheduling is tied to user credentials, and cannot be assigned to service principals or workspace identities, creating operational risk. 24:34	API Gaps for Organizational Apps: The creation and access management of organizational apps cannot be automated due to missing API support, forcing manual group assignments and limiting automation. 34:26	Lack of best practices, guidelines, or built-in support for unit testing and CI/CD in Fabric, forcing teams to create their own solutions and increasing maintenance burden 30:53	Gen2 Connector Limitations: They described issues with legacy data sources requiring Gen2 connectors, which forced manual redeployment in every environment and prevented parameterization of source and destination GUIDs, making deployments error-prone and labor-intensive. 22:54	Capacity and Performance Issues: They experienced capacity overloads (up to 29,000%) due to unoptimized queries (e.g., select star, missing conditions), causing the lakehouse to become unresponsive or copy jobs to hang and time out. There was also uncertainty about how smoothing works and how to manage capacity effectively. 39:14	Streaming Dataset Replacement: Lutz discussed the lack of an equivalent to Power BI streaming datasets in Fabric, which complicates migration and future planning. 15:50
Governance and Reporting: Additional custom solutions had to be built to provide visibility into workspace usage, permissions, and artifact refresh status, as native governance features were insufficient. 26:00	Permissions and Admin Rights: The interviewee noted challenges with limited admin rights in Fabric, requiring them to rely on IT for access to certain features and monitoring tools, which slowed down their workflow. 15:37	No robust solution for managing schema changes and deployments in lakehouse/warehouse environments, with unclear or missing best practices from Microsoft 38:50	Key Vault Integration: Managing secrets and retrieving them via Key Vault was described as more complex in Fabric compared to Synapse, requiring additional cloud engineering resources and coding, which added friction for less experienced developers. 36:48	Lakehouse Concurrency Issues: When more than three copy jobs ran in parallel, the lakehouse sometimes became unresponsive, with jobs timing out after long periods. 41:57	General lack of enterprise-level features in Fabric compared to Azure, causing hesitation to migrate critical solutions and leading to continued use of Azure for larger, business-critical projects. 38:52
Governance & Security Complexity: Challenges in configuring B2B sharing and semantic model deployment across tenants, with confusion and repeated setup steps for clients. 38:25	Granularity of Permissions: They expressed frustration with the lack of granular permission controls in Fabric, specifically wanting users to be able to refresh reports without having broader permissions to delete or move them. 42:29	Data lineage features in Fabric are limited and do not provide detailed table or attribute-level tracking, reducing transparency and traceability 50:20	Deployment Pipeline Issues: The deployment pipeline UI was described as unreliable, with frequent lockups and context switching problems that disrupted workflow and made environment comparisons difficult. 56:05	Performance and Data Explosion Issues: Nagesh mentioned that deploying large semantic models (e.g., 120GB) to the XMLA endpoint in Fabric caused the Vertipaq engine to "explode" in size, leading to severe performance problems (e.g., reports taking up to 30 minutes to load). They cannot tweak or control the engine as they could with SSAS or Azure Analysis Services, which is a significant limitation. 32:07	Lack of clear, actionable blueprints or success stories for migrating complex semantic models with security requirements to Fabric, making adoption decisions difficult.
Lakehouse Shortcuts and Security: Deploying lakehouse shortcuts is problematic, sometimes requiring removal and re-adding of shortcuts. Security is also a concern, with over-provisioning of permissions due to limitations in the current model, and uncertainty about the rollout of the unified security model. 21:10	Challenges with connection and ownership management: when a developer leaves, solutions tied to their user account can break, requiring significant manual intervention to restore functionality and access. 47:12	Security and governance controls are insufficient for some enterprise needs, such as column-level security and restricting user access to certain features or experiences 49:09	Data Activator Limitations: Data Activator could not handle dynamic, custom logic for Teams notifications required by customers, such as monitoring KPIs for specific time windows (e.g., current day plus two days). This forced them to use Power BI's built-in Teams notifications instead. 11:54	Event House Issues: Lutz noted that Event House consumes capacity units (CUs) even when idle, and ingestion is resource-intensive, which was unexpected. 14:37	Event Stream vs. KQL Ingestion: User found that using Event Stream for data ingestion added unnecessary compute and complexity, and debugging was difficult. He determined that direct ingestion into KQL was more efficient, but this requires workarounds and highlighted limitations in Fabric's streaming architecture. 26:45
Object Ownership Issues: When a developer leaves, items they own can stop working, and errors are not clear, making troubleshooting difficult. 50:56	Concerns about data exfiltration, especially users exporting large volumes to Excel, with limited ability to monitor, restrict, or audit these actions effectively. 2	Pipeline was missing some data and lack of proactive email alerts was there. Logs were difficult to correlate and messages were also generic.	Git Integration Issues: Robin described struggles with Fabric's git integration, noting that certain item types (like event streams and real-time dashboards) failed during deployment, and the process required multiple runs due to deployment order problems. Permissions also hindered product teams from running deployment pipelines as intended. 27:09	Difficulty controlling and splitting Fabric workloads by workspace, leading to risks of overconsumption, cost overruns, and lack of granular governance. 1	SQL Authentication Problems: There were intermittent authentication failures when vendors tried to connect to the Snowflake environment, sometimes resolved by retries or by vendors adding retry logic to their systems. 30:22
Technical Limitations and Roadblocks: He described		I ran into a pipeline issue in Microsoft Fabric due to an unexpected surge in data load. The problem was significant			

# How do we organize and prioritize all these issues?

... first, we map raw issues to our exiting [Fabric JTBD inventory](#)

## Pro-dev issue

Deployment and Git Integration Issues: The user described the deployment process as slow and manual, with limited automation. Git integration is not fully supported for deployments, leading to reliance on a point-and-click interface and manual workarounds.



## Main-JTBD

Design, implement, secure, optimize, and maintain data pipelines

## Sub-JTBD

Implement CI/CD & Automated deployment

Complexity and resource constraints made it challenging to implement and evaluate Fabric for the financial reporting scenario, resulting in project abandonment.



Develop and deploy data platform solutions

Design the POC

Workspace sprawl and capacity management issues, as developers create multiple workspaces for feature branches, raising concerns about governance and accidental resource overuse



Ensure data systems are available and performant

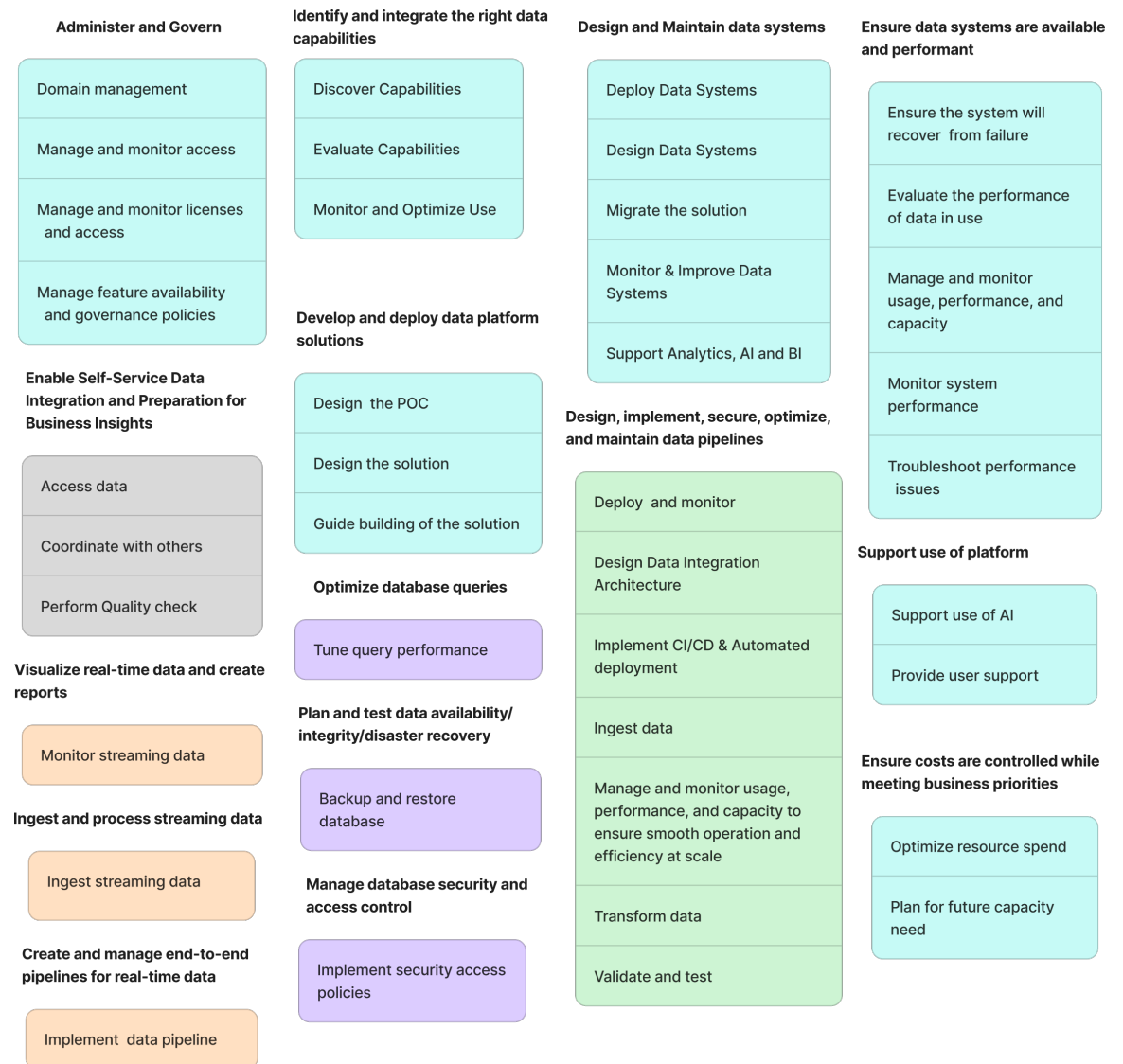
Manage and monitor usage, performance, and capacity

[Catalog of all raw issues mapped to JTBD](#)

The extracted issues mapped to 15 main jobs and 39 sub-jobs.

These are the ‘complaint zones’ – areas where developers are both **active and unhappy**

How unhappy? We did an opportunity analysis to figure that out.



# Opportunity analysis

Then, we conduct an opportunity analysis

JTBD with **high importance + low satisfaction = biggest opportunity**

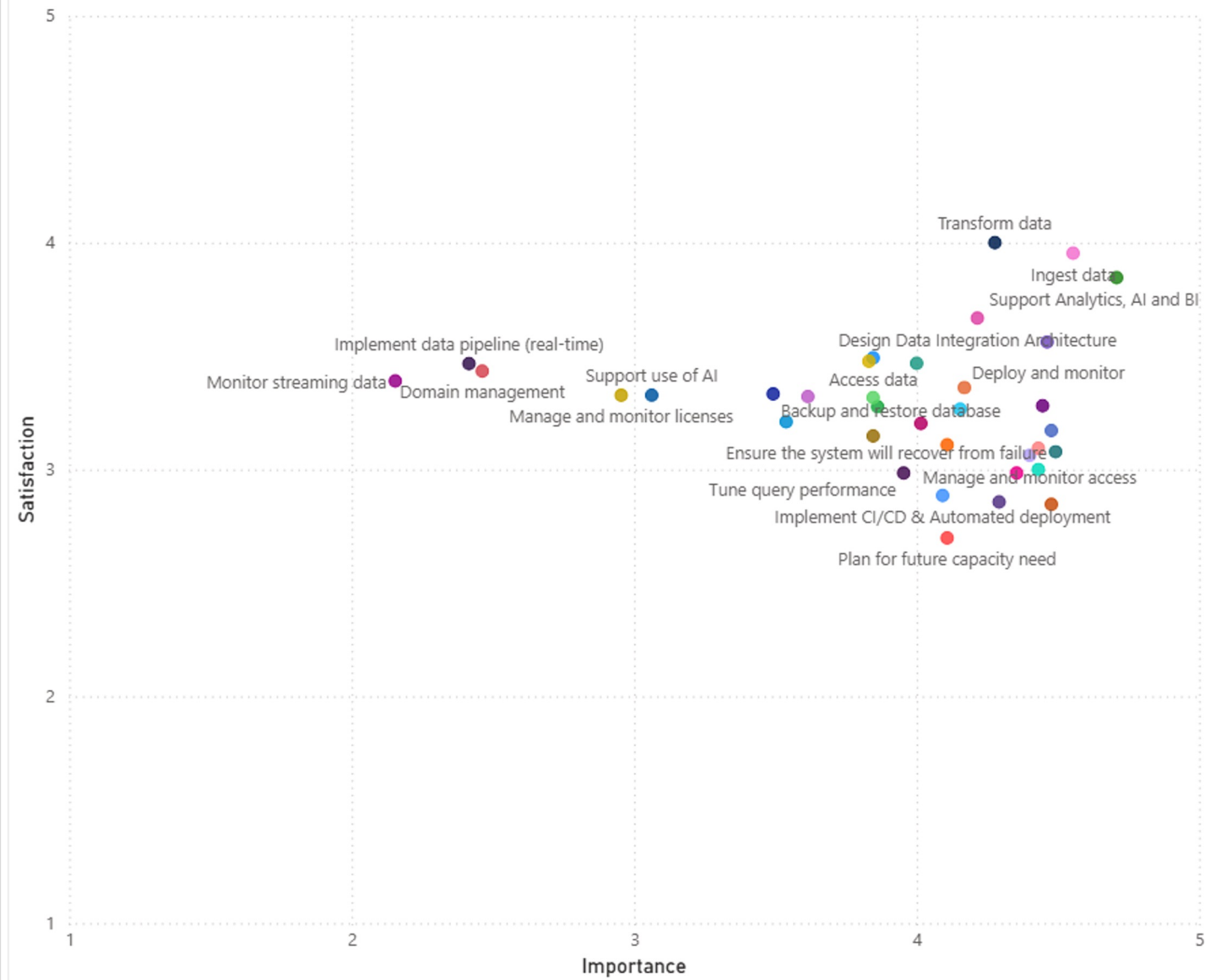
- How **important** is it to you to minimize the effort required to... [insert sub-job here]?
- How **satisfied** are you with the effort required to... [insert sub-job here]?

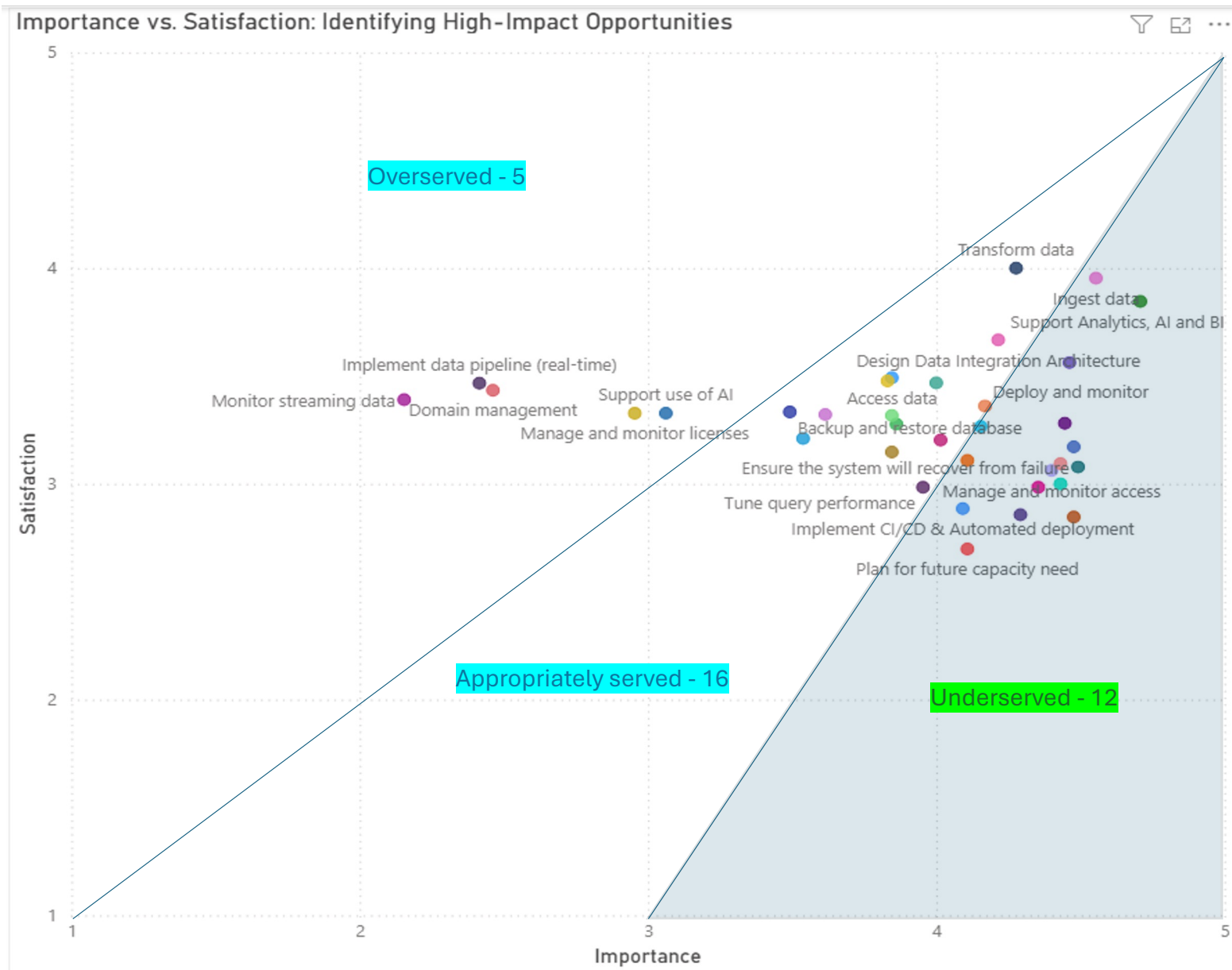
# Who did we survey?

## Things you should know about the sample.

- 113 users completed the survey through the CAT Fabric User Panel
- 32 users completed the survey using a vendor sourced panel
- 145 total respondents started the survey, 110 fit our profile and were included in the analysis
- Important to know: Questions were presented to users based on what activities they engaged in on Fabric. Therefore, not every question had all 110 responses.
- How do we account for the smaller sample size, overall?
  - We should have confidence given the QUALITY of the sample. We interviewed these folks and had prior engagements with them. That gives us confidence in generalizing from a smaller sample size (pitch for in-product survey tool here).
  - We limited analysis to questions with at least 60 responses to ensure confidence in generalizability. Exact +/- errors are difficult in this case, so specific values are not included.
  - Results from this study map very, very closely to prior summaries of our pro-developer engagements – giving us greater confidence in the findings
- Sample size is not large enough for demographic crosstabs (e.g., tenure on Fabric, weekly usage, etc.)

## Importance vs. Satisfaction: Identifying High-Impact Opportunities

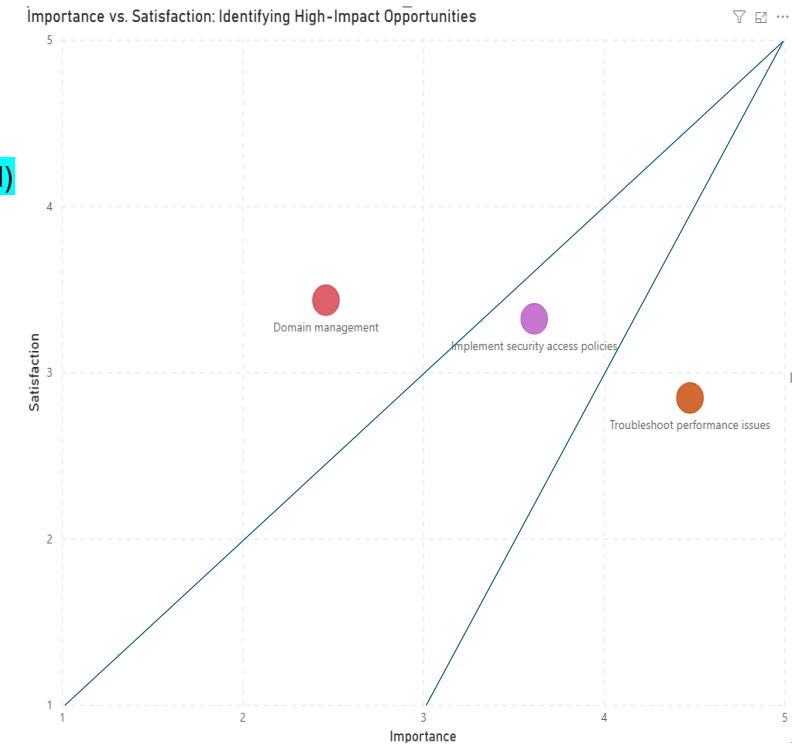




# Opportunity analysis

JTBD with high importance + low satisfaction = biggest opportunity

Sub-Job	Imp	Sat	Opportunity score
Troubleshoot performance issues	4.5	2.9	6.1 (underserved)
Implement security access policies	3.6	3.3	4.8 (appropriately served)
Domain management	2.5	3.4	3.2 (overserved)



## We identified 13 **Underserved** jobs:

### CI/CD & Deployment:

- Implement CI/CD & Automated deployment
- Deploy Data Systems

### Monitoring/troubleshooting:

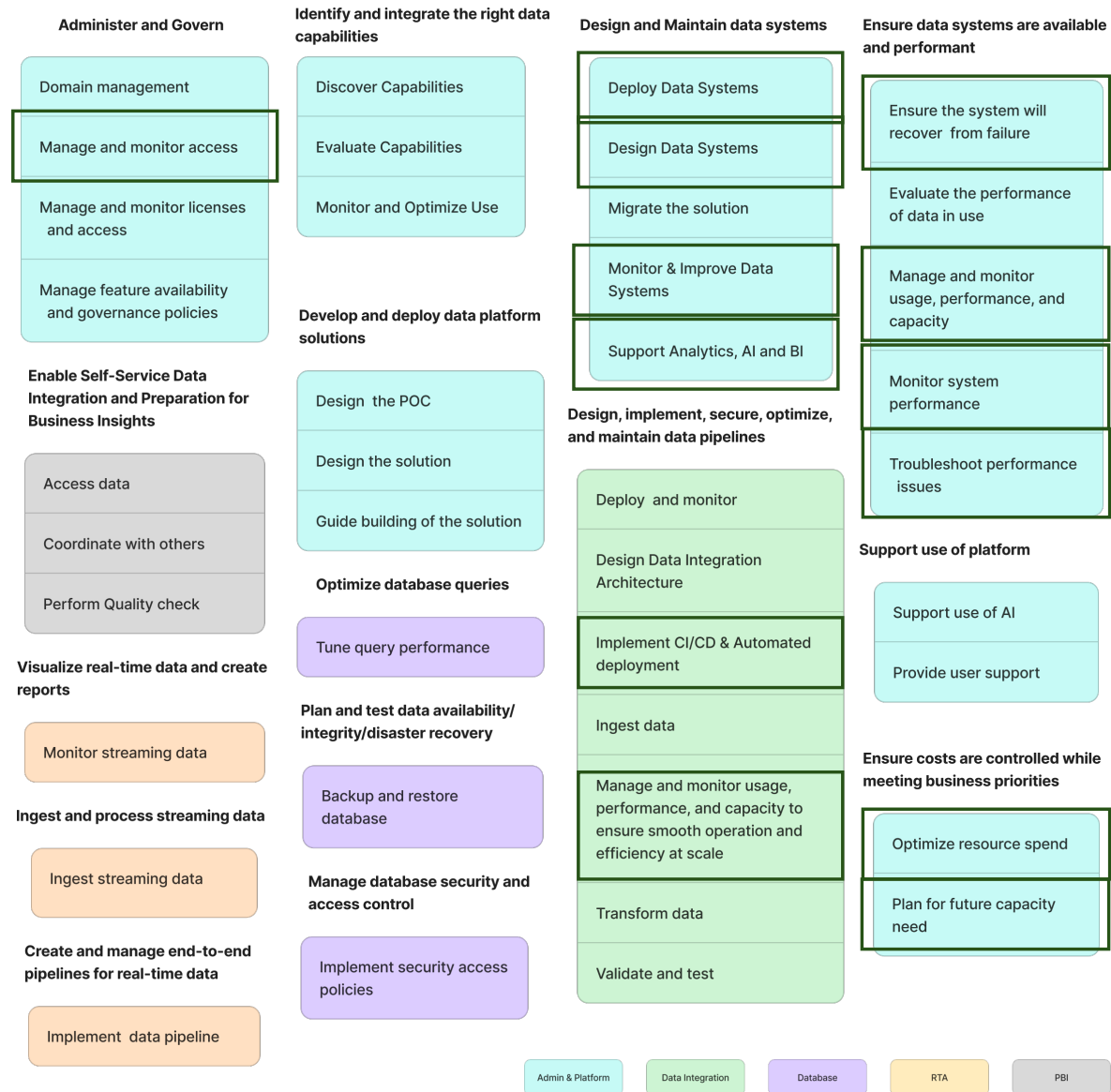
- Ensure the system will recover from failure
- Manage and monitor usage, performance, and capacity
- Monitor system performance
- Troubleshoot performance issues
- Monitor & Improve Data Systems
- Manage and monitor usage, performance, and capacity to ensure smooth operation and efficiency at scale

### Optimization, capacity planning:

- Optimize resource spend
- Plan for future capacity need
- Design data systems

### Access & Identity Management

- Manage and monitor access





# What should we do?

There were A LOT of issues raised across all of our engagements. This section describes the highest impact opportunities to improve professional developer experiences.

Each of the following slides highlights a priority area based primarily on the results of this study but also includes results from prior CAT Research where applicable.

We asked professional developers to select their #1 preferred improvement to Fabric across 35 possible improvements, divided into 6 categories. When a slide has a recommendation in red it is a **top ranked** priority based on this survey. There is more information on [improvement ranking methodology](#) in the appendix

- This is a top ranked pro-dev opportunity based on user ranking

# CI/CD & Deployment - Underserved and High priority

Challenges around deployment and CI/CD continue to be among the most frequently reported pain points for Fabric professional developers. Although we've made progress in closing gaps in this area, developers are letting us know that we're not done yet. Professional developers reported high importance and low satisfaction with CI/CD and deployment tools and provided us with multiple high priority areas for Microsoft to improve on.

## Underserved jobs:

- Implement CI/CD & Automated deployment
- Deploy Data Systems



## We should prioritize:

- Improve deployment pipeline speed, reliability, and artifact support
- Improve tools for cross-workspace deployments
- Improve support for creating connection strings using parameters
- Provide comprehensive CI/CD and DevOps documentation.
- Strengthen Git integration and branch management

## What the professional developers told us in interviews about CI/CD and deployment:

- **Strengthen CI/CD capabilities** by expanding support for automated deployments and reducing reliance on manual workarounds or custom Azure DevOps pipelines.
- **Enhance Git integration** with full feature parity (e.g., gitignore support, robust error handling) to streamline version control and collaboration.
- **Introduce environment management tools** for separation and parameterization across dev/test/prod, minimizing manual configuration and risk.
- **Optimize deployment pipelines** for speed, reliability, and full artifact support, enabling end-to-end automation.
- **Ensure preview features are Git-ready** before release to prevent technical debt
- **Automate deployment tasks** by adding configuration management and orchestration features, reducing repetitive manual steps.
- **Expand native functionality** to eliminate reliance on custom scripts and notebook utilities, improving efficiency and maintainability.

This is a top ranked pro-dev opportunity based on user ranking

# Monitor and Troubleshoot- Underserved and High priority

Fast iteration is critical for developers. Their productivity—and overall satisfaction with Fabric—depends on how quickly they can identify, diagnose, and resolve issues throughout the development lifecycle.

## Underserved jobs:

- Ensure the system will recover from failure
- Manage and monitor usage, performance, and capacity
- Troubleshoot performance issues
- Monitor & Improve Data Systems



## We should prioritize:

- Enhance monitoring speed, reliability, and alerting capabilities.
- Provide proactive pipeline alerts and success status.
- Improve end-to-end lineage to ensure data traceability.
- Un-silo monitoring tools
- Improve error messages and integrate resolution into error detection workflows

## What the professional developers told us in interviews about monitoring and troubleshooting:

- **Fragmented Logs:** Logs are scattered across multiple tools (Pipeline, Spark, Eventhouse), making root cause analysis slow and error-prone.
- **Alerting Gaps:** No scalable, centralized alerting for pipeline or notebook failures; alerts are unreliable, forcing teams to build custom notification systems.
- **Troubleshooting Challenges:** Issues like Spark memory failures, concurrency errors, and capacity overloads are hard to diagnose due to vague error messages and missing execution metrics.
- **Ownership Risks:** Artifacts tied to individual user accounts break when users leave.
- **Data Monitoring:** Concerns about data exfiltration and lack of monitoring for large exports.
- **Limited Visibility & Reporting:** Native tools lack dashboards, forcing teams to build custom solutions for usage, permissions, and artifact refresh tracking.

This is a top ranked pro-dev opportunity based on user ranking

# Optimization and Capacity Management- Underserved and High priority

Visibility into platform behavior is critical for developer success when building performant solutions with predictable expenses.

## Underserved jobs:

- Optimize resource spend
- Plan for future capacity need
- Design data systems

## What the professional developers told us in interviews about capacity planning and optimization on Fabric:

- **Capacity contention:** Multiple teams and jobs contend for the same capacity, so one heavy/inefficient workload can degrade or block others. Lack of isolation, queue backlogs, and cross-workspace interference make performance unpredictable and difficult to coordinate.
- **Difficult to predict capacity cost:**
  - Developers face delays enabling features due to concerns about capacity usage, security, and lack of product familiarity.
  - Gen2 connectors and Eventhouses consume excessive capacity—even when idle—prompting shifts to pipelines and notebooks for stability.
  - Multiple workspaces for feature branches complicate capacity planning and increase accidental resource overuse.
- **Difficulty controlling capacity overages:** Sudden, unexplained surges in capacity usage—even without changes in logic or data volume—create cost concerns and operational risk.
- **Capacity monitoring shortcomings:** Capacity health tools lack historical depth and fail to provide clear root-cause insights, forcing manual investigation.



## We should prioritize:

- Expose capacity costs for all activities
- Enable flexible cost breakdowns for compute charges, allowing organizations to view and allocate costs by their preferred dimensions.
- Continue development on tools to guard against cost overages

## Access & Identity Management - Underserved and High priority

Challenges around managing access and identity are common when working with enterprise customers and are a recurring theme with professional developers as well. There's no easy way to manage identities or automate permissions, security controls are inconsistent, and visibility is limited.

### Underserved jobs:

- Manage and monitor access

### What the professional developers told us in interviews about capacity planning and optimization on Fabric:

- **Lack of robust identity/connection management:** Without service accounts or managed identity support, processes (e.g., scheduled pipelines or shortcuts) need to run under individual user credentials
- **Fragmented Access Control:** Permissions for SQL endpoints and other artifacts are manual and not integrated with CI/CD, creating gaps.



### We should prioritize:

- Enable service accounts
- Provide tools to automate identity and access management

This is a top ranked pro-dev opportunity based on user ranking

# Prior Professional Developer Program Reports

- The results from this quantitative Professional Developer Opportunity study map closely to the themes observed in prior interview summaries – **giving us high confidence in the prioritization of the issues they've shared with us**
- Prior professional developer reports:
  - Trending themes from the first 50 interviews
    - [Trending Insights from the Professional Developer Connect Program](#)
  - Themes from Cohort 1 (Interviews conducted July 31-Aug 14)
    - [Pro Dev Connect \(Cohort 1 Study\)](#)
  - Themes from Cohort 2 (Interviews conducted Aug 15-Aug 27)
    - [Pro Dev Connect \(Cohort 2 Study\)](#)

# Next Steps / Call to action

- Review deck for insights relevant to you, how they will or are fitting into your plans
- Identify follow-up participants to learn more – engage with CAT research team for connections (EV connect app link)
  - Explore the [Fabric User Panel](#)
  - Review other [CAT programs](#) facilitating customer engagements
- Talk with YOUR researcher!

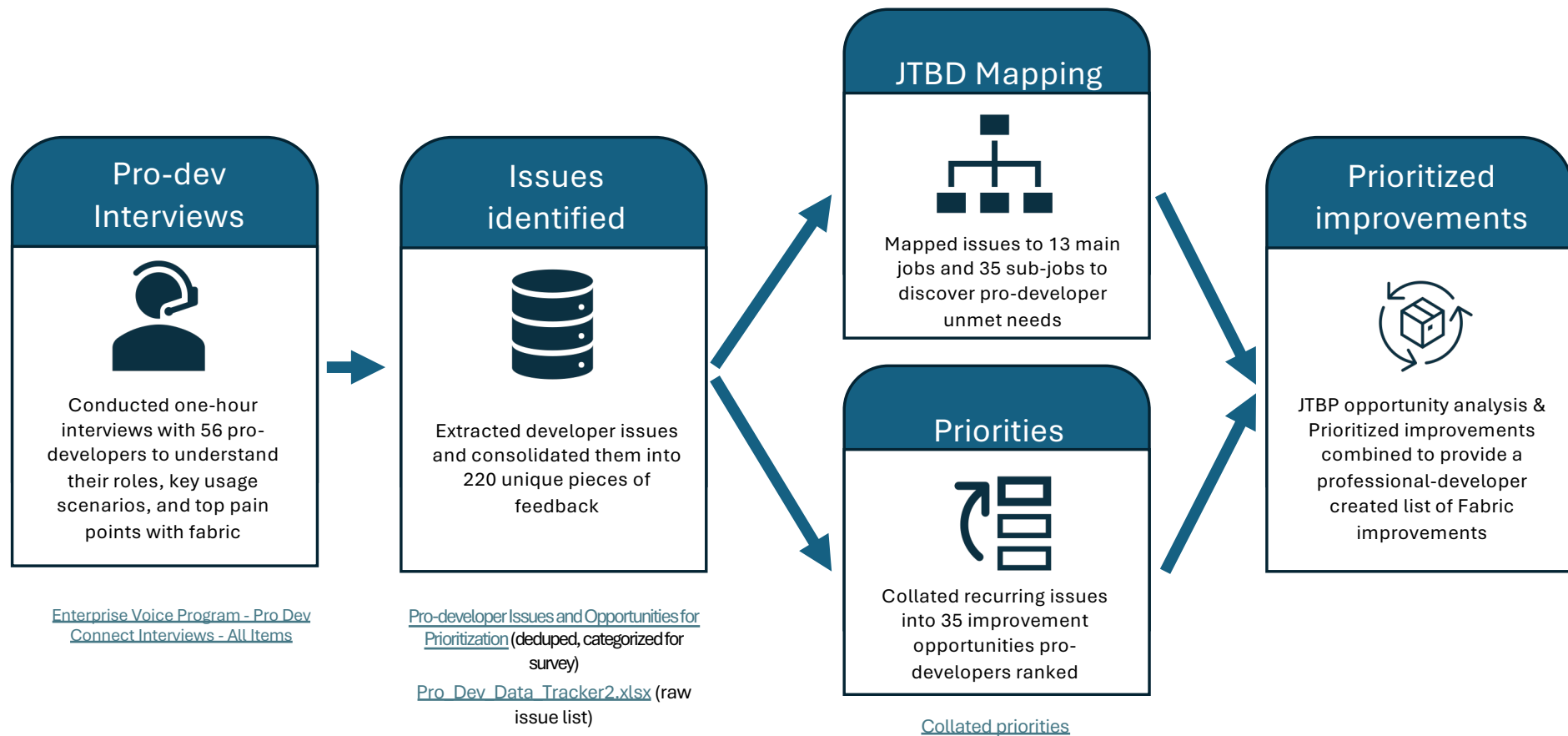
# Issue Tracking

		Team	UR	Semester priority	Has a plan	Requires additional research
<b>CI/CD Deployment:</b>	<b>Aviv</b>					
Improve deployment pipeline speed, reliability, and artifact support	Aviv	DI	Dawn	?		
Improve tools for cross-workspace deployments	Aviv	DI	Dawn	Platform has this listed as below the cut line		Fabric workspace variables are listed
Improve support for creating connection strings using parameters	Wee Hyong (?)	DI	Dawn	Yes		
Provide comprehensive CI/CD and DevOps documentation.	Aviv	DI	?	yes		Has this ever been identified as an issue?
Strengthen Git integration and branch management	Aviv	DI	Dawn	yes, but workloads need to do the integration - th		Planning doc says git is all green
<b>Monitor and Troubleshoot:</b>	<b>Arthi (Capacity)/ Hayden (Job)</b>					
Enhance monitoring speed, reliability, and alerting capabilities.	Arthi/Hayden	Platform	Bruce	yes		Alerts have come up in multiple projects
Provide proactive pipeline alerts and success status. (scheduling)	Hayden (scheduling)	Platform	Bruce (scheduling)	yes for plat - proactive job alerting; DI Pro-dev pipeline		Bruce
Un-silo monitoring tools	Arthi/Hayden	Platform	Bruce	yes, capacity team bringing capacity monitoring		Bruce
Improve error messages and integrate resolution into error detection workflow	Scott Gerlach/everyone!	Platform (Laurer)	Bruce	yes, in general. platform working on error message framework, which informs workload		
<b>Optimization and Capacity Management-</b>	<b>Arthi</b>					
Expose capacity costs for all activities	Arthi	Platform	Bruce	no, not this semester. Working on getting the data		research on , what to expose and how
Enable flexible cost breakdowns for compute charges, allowing organizations	Arthi	Platform	Bruce	?		what and how do users want to view
Continue development on tools to guard against cost overages	Arthi	Platform	Bruce	yes - surge protection v2, integration with capacity		support sp.v2 planning
<b>Access &amp; Identity Management</b>	<b>Adi</b>					
Improve end-to-end lineage to ensure data traceability and governance	Adi	Platform	Avishag			
Enable service accounts	Adi, Ravs, Wee Hyong (DI)	Platform	Avishag			
Provide tools to automate identity and access management	Adi (OL gov)/Arthi (workspace/identity)	Platform	Avishag/Bruce	team is working on 'item roles ' and 'item identity'		better understanding of what users r

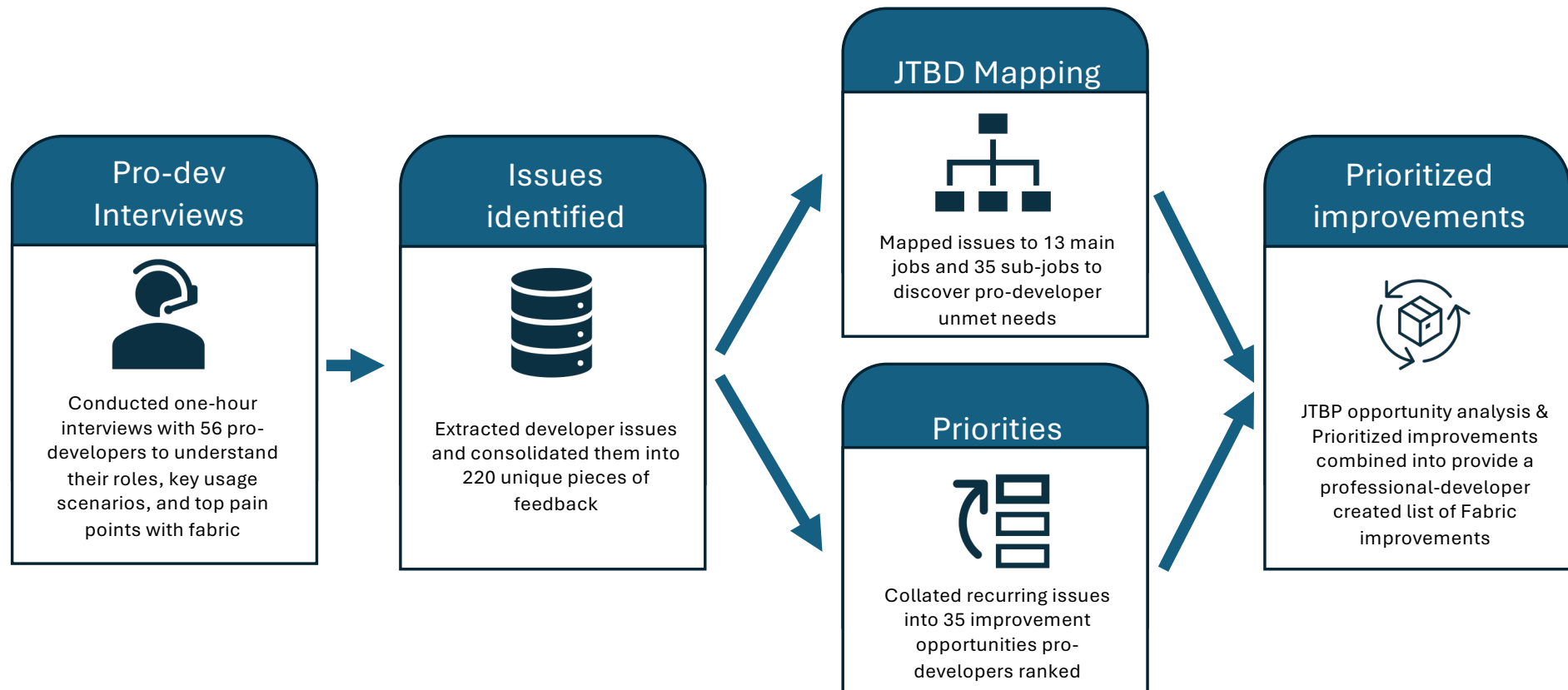
[Prodev Opportunity Tracker.xlsx](#)

# Appendix

# Method



# Method



Main Job

Sub Job



Enable Self-Service Data Integration and Preparation for Business Insights	Access data
Plan and test data availability,integrity, disaster recovery	Backup and restore database
Design, implement, secure, optimize, and maintain data pipelines	Deploy and monitor
Design and Maintain data systems	Deploy Data Systems
Design, implement, secure, optimize, and maintain data pipelines	Design Data Integration Architecture
Design and Maintain data systems	Design Data Systems
Identify and integrate the right data capabilities	Discover Capabilities
Administer and Govern	Domain management
Ensure data systems are available and performant	Ensure the system will recover from failure
Identify and integrate the right data capabilities	Evaluate Capabilities
Ensure data systems are available and performant	Evaluate the performance of data in use
Design, implement, secure, optimize, and maintain data pipelines	Implement CI/CD & Automated deployment
Create and manage end-to-end pipelines for real-time data	Implement data pipeline (real-time)
Manage database security and access control	Implement security access policies
Design, implement, secure, optimize, and maintain data pipelines	Ingest data
Administer and Govern	Manage and monitor access
Administer and Govern	Manage and monitor licenses
Ensure data systems are available and performant	Manage and monitor usage, performance, and capacity
Design, implement, secure, optimize, and maintain data pipelines	Manage and monitor usage, performance, and capacity to ensure smooth operation and efficiency at scale
Administer and Govern	Manage feature availability
Design and Maintain data systems	Migrate the solution
Design and Maintain data systems	Monitor & Improve Data Systems
Identify and integrate the right data capabilities	Monitor and Optimize Use
Visualize real-time data and create reports	Monitor streaming data
Ensure data systems are available and performant	Monitor system performance
Ensure costs are controlled while meeting business priorities	Optimize resource spend
Enable Self-Service Data Integration and Preparation for Business Insights	Perform Quality check
Ensure costs are controlled while meeting business priorities	Plan for future capacity need
Support use of platform	Provide user support
Design and Maintain data systems	Support Analytics, AI and BI
Support use of platform	Support use of AI
Design, implement, secure, optimize, and maintain data pipelines	Transform data
Ensure data systems are available and performant	Troubleshoot performance issues
Optimize database queries	Tune query performance
Design, implement, secure, optimize, and maintain data pipelines	Validate and test

## Mapping top opportunities to top priorities



Brief description of opportunity analysis

**Directly Link User Pain Points to Strategic Product Opportunities:**

By mapping real-world challenges to jobs, teams can focus on improvements that will have the highest impact on user productivity and satisfaction.

**Drive Measurable Improvements:**

Quantifying unmet needs helps teams track progress and measure the impact of changes over time.

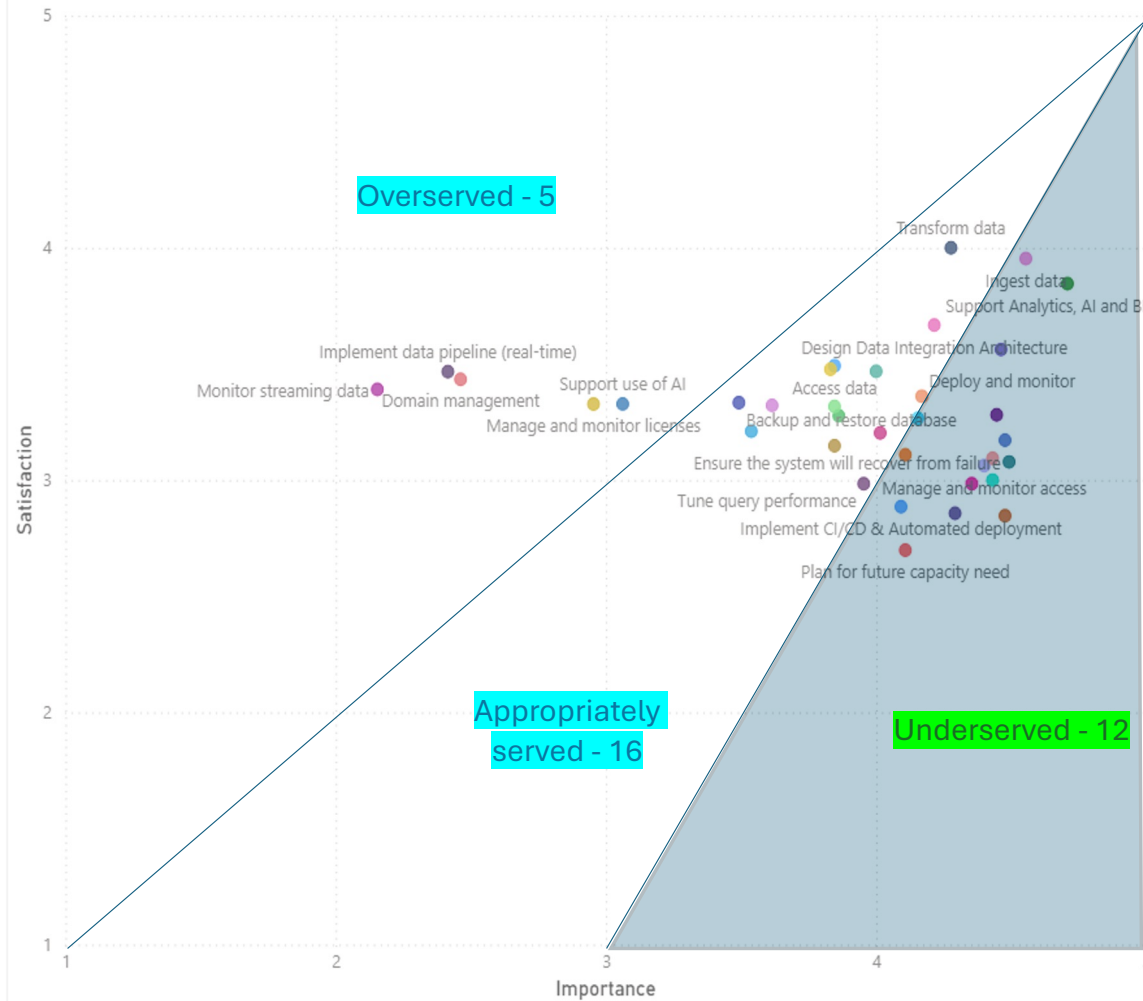
**Empower Leadership to Make Data-Driven Decisions:**

The analysis provides clear recommendations for where to invest resources, ensuring that product changes align with the most critical user needs.

**Create a Shared Understanding Across Teams:**

JTBD analysis builds consensus around what matters most to users, guiding cross-functional teams toward common goals.

Importance vs. Satisfaction: Identifying High-Impact Opportunities



Underserved:

- Monitor, Troubleshoot, optimize

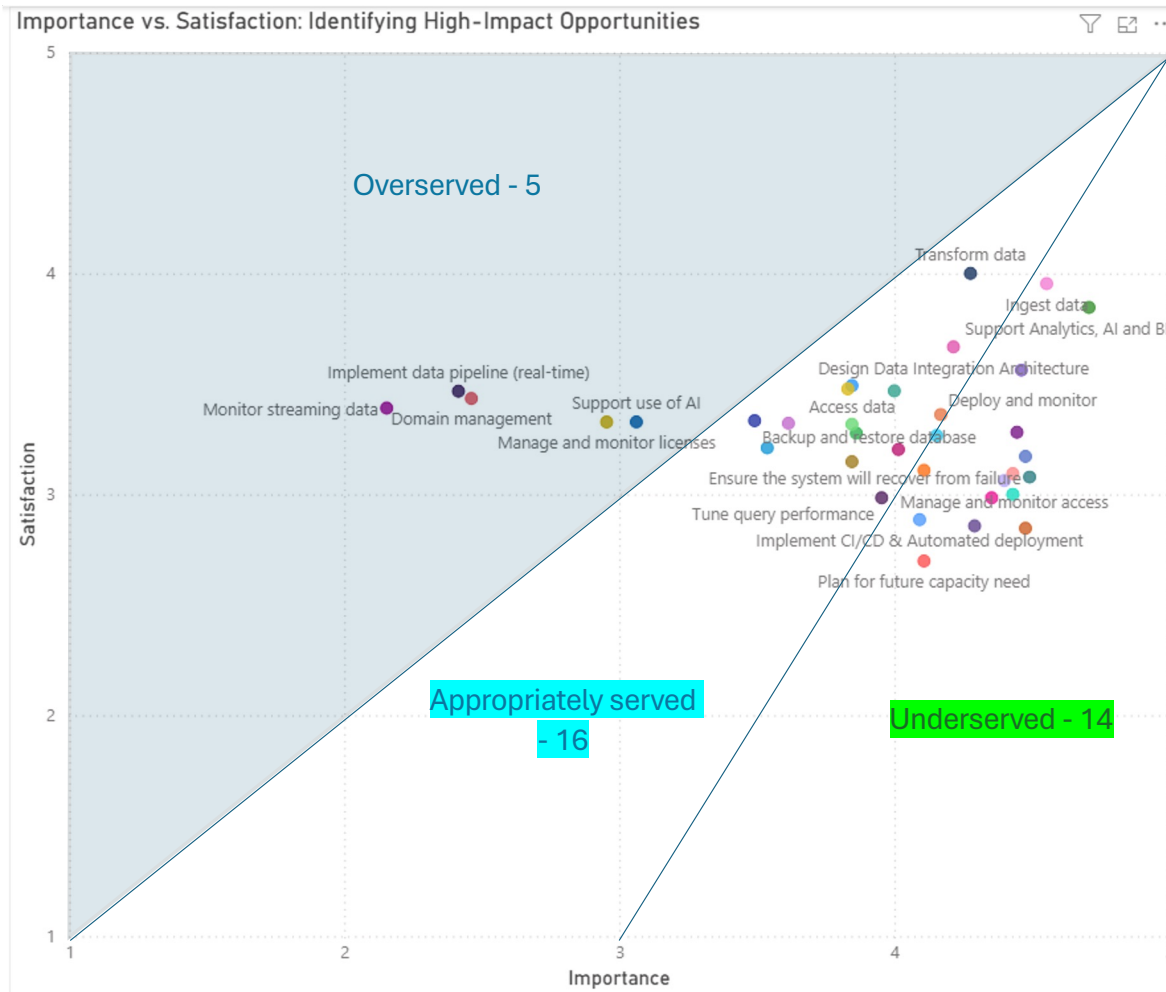
- Deployment, CI/CD

Main Job	Sub Job
Ensure data systems are available and performant	Troubleshoot performance issues
Ensure data systems are available and performant	Ensure the system will recover from failure
Ensure costs are controlled while meeting business priorities	Optimize resource spend
Design, implement, secure, optimize, and maintain data pipelines	Manage and monitor usage, performance, and capacity to ensure smooth operation and efficiency at scale
Ensure data systems are available and performant	Manage and monitor usage, performance, and capacity
Design and Maintain data systems	Monitor & Improve Data Systems
Ensure data systems are available and performant	Monitor system performance
Ensure costs are controlled while meeting business priorities	Plan for future capacity need
Design, implement, secure, optimize, and maintain data pipelines	Implement CI/CD & Automated deployment
Administer and Govern	Manage and monitor access
Design and Maintain data systems	Deploy Data Systems
Design, implement, secure, optimize, and maintain data pipelines	Deploy and monitor
Design and Maintain data systems	Support Analytics, AI and BI
Design and Maintain data systems	Design Data Systems



## Appropriately served - 16

Area	Main Job	Sub Job
Databases	Optimize database queries	Tune query performance
Data Integration	Design, implement, secure, optimize, and maintain data pipelines	Ingest data
Fabric	Identify and integrate the right data capabilities	Monitor and Optimize Use
Fabric Platform	Ensure data systems are available and performant	Evaluate the performance of data in use
Data Integration	Enable Self-Service Data Integration and Preparation for Business Insights	Perform Quality check
Databases	Plan and test data availability, integrity, disaster recovery	Backup and restore database
Data Integration	Design, implement, secure, optimize, and maintain data pipelines	Design Data Integration Architecture
Data Integration	Design, implement, secure, optimize, and maintain data pipelines	Validate and test
Fabric Platform	Support use of platform	Provide user support
Data Integration	Design, implement, secure, optimize, and maintain data pipelines	Transform data
Fabric Platform	Administer and Govern	Manage feature availability
Data Integration	Enable Self-Service Data Integration and Preparation for Business Insights	Access data
Databases	Manage database security and access control	Implement security access policies
Fabric Platform	Identify and integrate the right data capabilities	Evaluate Capabilities
Fabric Platform	Design and Maintain data systems	Migrate the solution
Fabric Platform	Identify and integrate the right data capabilities	Discover Capabilities



## Overserved - 5

Area	Main Job	Sub Job
Fabric Platform	Administer and Govern	Manage and monitor licenses
	Create and manage end-to-end pipelines for real-time data	Implement data pipeline (real-time)
Business Real Time Intelligence	Visualize real-time data and create reports	Monitor streaming data
Fabric Platform	Support use of platform	Support use of AI
Fabric Platform	Administer and Govern	Domain management

# Ranking Fabric Improvements

- The second part of the study had professional developers rank order Fabric improvements
- Improvement options were generated from the raw feedback we received from pro-devs
- Feedback was deduped and consolidated into 6 categories with 35 total improvement to rank

# Converting issues to improvement priorities

Generating improvement priorities, going from original pro-dev statement to improvement statement for ranking

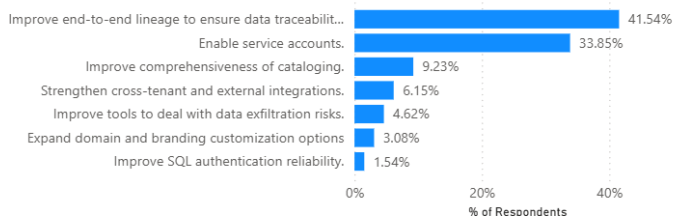
	Original problem statement	Re-worded improvement statement for ranking
Performance & Monitoring	Spark environment startup times are long (3–5 minutes).	Reduce Spark environment startup times
	Workspace proliferation worsens governance and costs.	Simplify workspace management to reduce sprawl and cost.
	Monitoring is slow, unreliable, lacks filtering; alerts untrustworthy.	Enhance monitoring speed, reliability, and alerting capabilities.
Documentation & Support	Documentation for CI/CD and DevOps is fragmented	Provide comprehensive CI/CD and DevOps documentation.
	Support channels slow; informal networks more useful.	Improve official support speed and effectiveness.
Pipelines & Data Quality	Pipelines miss data without alerts; may falsely show “success.”	Provide proactive pipeline alerts and accurate success status.
	Spark jobs fail intermittently with vague errors.	Improve Spark job stability and error transparency.
CI/CD & Deployment	Deployment pipelines are slow, error-prone, and incomplete.	Improve deployment pipeline speed, reliability, and artifact support.
	Git integration immature (branch issues, unreliable for large workspaces).	Strengthen Git integration for scalability and branch management.

# Pro-dev priorities

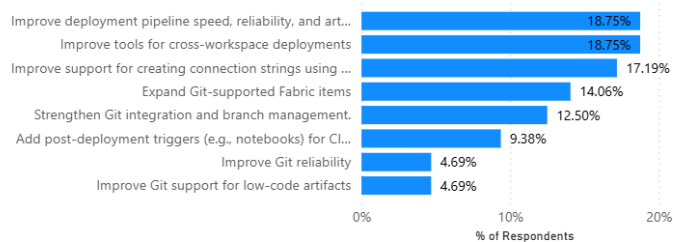
Ranking is conducted within each of the 6 categories. There is no overall ranking.

- In this section we'd like you to help us **prioritize future improvements to Fabric**. Please select one item from the list that is **most important to you**.

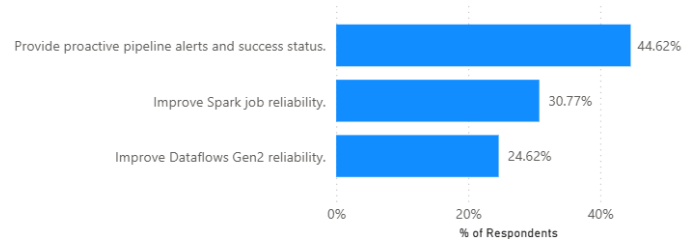
## Governance, Security & Access



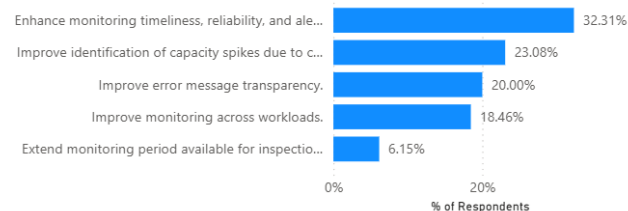
## CI/CD & Deployment



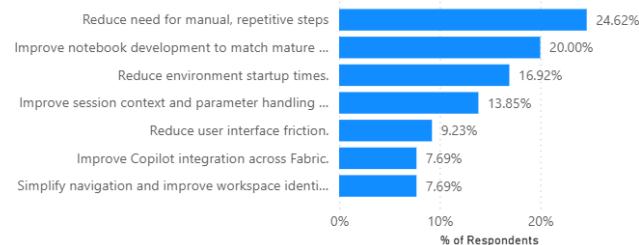
## Pipelines



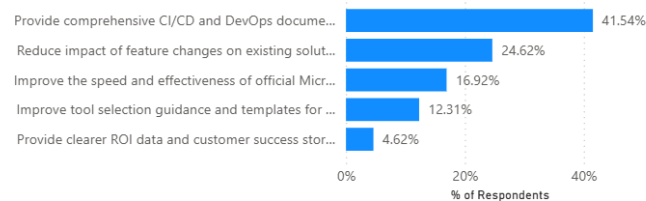
## Performance & Monitoring



## Development Environment



## Adoption, Support & Training



# Connecting JTBD opportunities and priorities

JTBD opportunity ( ↑ importance, ↓ satisfaction)

Top-ranked improvement priorities

## Monitor

- Manage and monitor usage, performance, and capacity.
- Monitor & Improve Data Systems.
- Monitor system performance.
- Manage and monitor access.
- Troubleshoot performance issues.



## Monitoring, troubleshooting, optimizing

- Enhance monitoring speed, reliability, and alerting capabilities.
- Provide proactive pipeline alerts and success status.
- Enable service accounts.

## Deployment, CI/CD

- Manage and monitor usage, performance, and capacity to ensure smooth operation and efficiency at scale.
- Manage and monitor usage, performance, and capacity.
- Monitor & Improve Data Systems.
- Monitor system performance.
- Manage and monitor access.



## Deployment, CI/CD

- Improve deployment pipeline speed, reliability, and artifact support.
- Improve tools for cross-workspace deployments.
- Improve support for creating connection strings using parameters.
- Provide comprehensive CI/CD and DevOps documentation.

## Development Environment

- Reduce need for manual, repetitive steps.