EXECUTIVE SUMMARY

Embedding **Fabric Everywhere**: Making Fabric Accessible Beyond the Portal

Fabric is powerful inside the portal, but increasingly work happens outside it, in CI/CD pipelines, automation scripts, and AI agents. Platforms are externalizing their capabilities to **meet users where they are**. "Embedding Fabric Everywhere" means exposing Fabric's full capabilities programmatically, so developers can automate and AI agents can operate, for example, building a data pipeline from ChatGPT or managing deployments from VS Code. We propose building on our team's strengths (automation, embedding, platform work, MCP infra) to lead this shift.

This document outlines a strategic direction. The following slides walk through the scenarios, architecture, and proposed starting points for discussion.

The Market Is Shifting: Platforms Must Be Embeddable to Stay Relevant

Three macro trends are reshaping how users interact with platforms — and what they expect from Fabric.

Search → Chat

Users ask AI instead of clicking links. AI summaries replace traditional browsing.

58.5% of Google searches end in zero clicks (SparkToro 2024)

Static UIs → Agent UIs

AI agents generate interfaces on-the-fly. Users interact through tools rendered inline.

70% of API developers are MCP-aware (Postman 2025)

Portals → Embedded

Users expect platforms to come to them — in their IDE, chat, or CI/CD pipeline.

82% of orgs follow API-first strategies (Postman 2025)

Bottom line: Platforms that only live in a browser portal risk becoming invisible.

OUR VISION

Fabric Capability Should Be Accessible from Anywhere

Today "Fabric Embed" means embedding a report. Tomorrow it means embedding any Fabric experience or capability.

> *”By 2028, every Fabric capability should be embeddable, invocable, and composable — by humans, scripts, and AI agents."*

WHAT THIS MEANS

- Every capability accessible via API
- Key capabilities accessible via MCP (for AI agents)
- Every workflow scriptable via CLI
- Full operations possible outside the portal (portal remains the primary UI)

WHAT THIS LOOKS LIKE

- A developer troubleshoots a pipeline from VS Code (no portal visit needed)
- A ChatGPT user provisions a workspace in-place with a single request
- An autonomous CI/CD agent deploys and monitors heedlessly
- A citizen developer builds a data pipeline inside Copilot

Bottom line: Fabric meets users where they are.

Introducing AI Agents as a New Persona / Consumer Type

Fabric now serves two distinct consumer types, each with different needs.

Human Users

Data engineers, analysts, admins who interact via portal, IDEs, chat, Teams.

- Also build tools to extend their reach: scripts, CI/CD pipelines, Terraform, PowerShell
- Need: rich experiences + complete, reliable APIs for automation

AI Agents

Software agents that discover and invoke Fabric operations via MCP.

- Interactive agents: Human + AI copilot working together (e.g., Copilot in VS Code)
- Autonomous agents: Operate independently — monitoring, deploying, governing
- Need: discoverable tools, proper identity (SPNs), scoped permissions, audit trails

---

Bottom line: AI agents deserve first-class treatment, with their own identity, permissions, and audit trails.

Three Layers to Enable the Vision

Our strategy invests in three layers. Each layer brings unique standalone value, and combined they enable our vision.

Agentic Layer — AI-Powered Experiences

MCP Apps*, autonomous agents, customer MCP tools. Enables AI agents to operate Fabric intelligently.

Programmatic Layer — Classic Automation

CI/CD pipelines, scripts, Terraform, scheduled jobs. Delivers immediate value today (even without AI).

Foundational Layer — Infrastructure

APIs, CLI, MCP Server, SDKs, identity, monitoring. The reliable plumbing everything is built on.

Bottom line: Foundation enables Programmatic enables Agentic — but each layer delivers standalone value.

MCP Apps: Interactive UI components served by MCP servers, rendered inline in VS Code, Claude, ChatGPT — e.g., troubleshooting dashboards, pipeline builders.

What This Looks Like: From Minutes to Seconds, From Portal to Anywhere

The same three layers power different levels of automation — from human-guided to fully autonomous.

Developer Troubleshooting from VS Code

Pipeline fails → asks Copilot "Why did this fail?" → agent pulls logs via MCP → identifies memory issue → suggests fix → applies it → re-runs pipeline.

Result: 4 minutes vs 20+ minutes navigating the portal.

Business User Creating a Report from Claude

"Create a Q4 marketing report" → agent discovers semantic models → suggests layout → user refines → creates report via API → schedules refresh.

Result: Never opened the Fabric portal.

Autonomous CI/CD Agent in GitHub Actions

PR merged → agent deploys to staging → runs validation → tests fail → auto-rollback → posts summary to PR.

Result: Zero human intervention required.

Bottom line: Different users, different tools, same Fabric underneath. Fabric meets them where they are.

WHY WE ARE THE RIGHT TEAM FOR THIS

We own the stack this vision requires

Our team have both the foundational infrastructure and the externalization surface.

WHAT WE OWN

- **Embed:** Embedding Fabric items in external apps → expanding to embedding full experiences
- **APIs & Dev Tools:** Public APIs, CLI, SDKs, Terraform — the foundation layer
- **MCPs:** Local/Remote MCP + MCP infra — agent connectivity
- **Platform DNA:** As part of Fabric Platform, we know how to work with other Fabric teams and enable them. We've built that muscle.

THREE PARALLEL STREAMS OF WORK

- **Foundation:** APIs, CLI, SDKs, Identity — every API completed = one more thing agents can do
- **Programmatic:** CI/CD, Terraform, Scripting — every workflow automated = one less portal dependency
- **Agentic:** MCP Platform, MCP Apps — every tool published = one more surface where Fabric lives

---

Bottom line: We're the infrastructure team that enables everyone else and externalizes Fabric beyond the portal.

Proposed Starting Points: Foundation First, Then Prove Value End-to-End

These are examples of where we could start — pending alignment on vision and priorities.

FOUNDATION

API Gaps & Throttling

Close the top API gaps. Fix throttling for agent-scale traffic.

Prerequisite for everything else — scripts, agents, CI/CD all depend on complete, reliable APIs.

FOUNDATION

Internal API Tooling

Build tools that help other Fabric teams complete their APIs faster.

As the platform team, we accelerate the entire Fabric ecosystem — not just our own surface.

PROGRAMMATIC

CLI v2 + Scripting

The composable execution backbone for both programmatic and agentic layers.

Open-source, extensible CLI framework. Powers scripts, CI/CD, and AI agent actions.

AGENTIC

Troubleshooting MCP App

Proof-of-concept of the three-layer model, end-to-end.

Grounded in #1 CAT pain point (opp score 6.1). Demonstrates the full vision in one tangible experience.

This can be eventually expanded to include additional tools and experiences to tackle the top pains of code-first **pro-developers** in Fabric. We will enable and guide other teams to build their MCP apps as well.

We want first to align on vision. Then we scope work, commit resources and detail timelines.

MORE DETAILED SLIDES

"Embedding" Fabric Everywhere

A Strategy for the Agentic Era

Fabric Embed and Automation Team

February 2026

WHERE WE'RE HEADED

By 2028, Fabric is not just a portal you visit — **it is the data platform embedded in every tool, workflow, and agent your customers already use.**

- Every capability has an API
- Key capabilities accessible via MCP
- Every workflow scriptable via CLI
- AI agents operate Fabric natively
- Customers build on top of Fabric
- Full operations possible outside the portal

The best platforms won't ask users to come to them. They'll go to where the users are.

From Five Directions to One Story

In our last discussion, we explored 5 strategic directions. We believe they connect into a stronger, unified narrative.

1	**Portal Scripting**	Automate click-ops without leaving Fabric
2	**VS Code Experiences**	Meet developers in their IDE with sandboxing
3	**Functions as MCP Tools**	Expose business logic to AI agents
4	**Externalizing Agentic Capabilities**	External agents get internal Copilot power
5	**Blueprint Expansion**	IaC + policies + AI-assisted provisioning

The directions aren't removed — they're contextualized within a larger strategy that explains **why they matter** and **how they connect**.

Three shifts reshaping how people work

Search is becoming Chat

Users ask AI instead of clicking links. AI summaries replace traditional browsing.

58.5% of Google searches now end in zero clicks (SparkToro 2024)

Static UIs are becoming Agent UIs

AI agents generate on-the-fly interfaces. Users interact through tools like MCP Apps.

70% of API developers are MCP-aware (Postman 2025)

Portals are becoming Embedded

Users expect platforms to come to them — in their IDE, chat, or CI/CD pipeline.

82% of orgs follow API-first strategy (Postman 2025)

Platforms that only live in a browser portal risk becoming invisible.

What This Means for Fabric

Fabric is a powerful **SaaS platform** — and that brought real strengths in simplicity and reach. But today, not all Fabric capabilities are accessible outside the portal.

Not all UI actions have API equivalents. Agent and headless operation aren't yet fully supported.

> **To stay relevant in an AI-mediated world, Fabric needs to become fully embeddable and automatable — meeting users wherever they work.**

76%
of developers use or plan AI coding assistants

$52.6B
AI agent market by 2030 (46% CAGR)

89%
of API devs use generative AI daily

NPS 12
heavy Fabric users — least satisfied

OUR THESIS

Every Fabric capability should be **embeddable, invocable, and composable** — by humans and AI agents.

A developer troubleshoots a pipeline from VS Code — no portal visit needed

A ChatGPT user provisions a workspace in-place with a single request

An autonomous CI/CD agent deploys and monitors headlessly

A citizen developer builds a data pipeline inside Copilot

We Already Own the Building Blocks

We own both the foundational infrastructure and the embedding surface.

Fabric Embed
We embed Fabric items into external apps today. This vision extends it to embed any experience.

Public APIs
We own the foundational layer — the APIs everything is built upon. 110K MAU, 13% MoM growth.

Fabric MCP (Local + Hosted)
We own agent infrastructure — Local MCP (GA at FabCon March '26) and Remote/Hosted MCP.

CLI
Composable execution layer for scripts, agents, and power users. 168K downloads.

SDKs + Terraform
.NET SDK GA, Python SDK in preview, Terraform provider — developer tools ecosystem.

MCP Platform
We provide the MCP infrastructure for additional Fabric teams to build their MCP tools and apps.

Our team owns both the infrastructure (APIs, CLI, SDKs) and the embedding surface (Embed, MCP, Control Plane).

Two Types of Consumers That Use Fabric

In the agentic era, platforms serve both humans and AI agents — each with different needs and interaction patterns.

Human Users

Portal, IDE, Chat, Teams, CI/CD

Data engineers, analysts, and admins who interact through UI, conversational surfaces, and automation scripts.

Humans also build scripts and pipelines — CI/CD, Terraform, PowerShell — as tools to extend their reach.

TOOLS THEY USE

Portal · VS Code · CLI · CI/CD · Terraform · Scripts

AI Agents

A new standalone consumer

Software agents that discover and invoke Fabric operations via MCP — with their own identity, permissions, and audit trails.

Interactive
Human + AI copilot working together in conversation (e.g., Copilot in VS Code)

Autonomous
Operates independently — monitoring, deploying, governing (e.g., CI/CD agent)

The same foundational infrastructure (APIs, CLI, MCP) serves both — but AI agents need additional capabilities like identity, governance, and tool discovery.

Three Layers That Make It Work

Each layer delivers value on its own — and enables the layers above it.

Agentic Layer — AI-Powered Experiences

AI agents operate Fabric intelligently

- MCP Apps *
- Autonomous Agents
- Customer MCP Tools

powered by

Programmatic Layer — Classic Automation

Scripts, CI/CD, IaC — immediate value today

- Provisioning
- CI/CD & Deployment
- Access & Governance
- Bulk Operations
- Monitoring
- Scheduling

powered by

Foundational Layer — Infrastructure

The complete, reliable plumbing everything calls into

- REST APIs
- CLI
- MCP Server
- SDKs
- Identity and Governance
- Monitoring APIs

Key Insight

Each layer delivers value on its own — you don't need to wait for agents to benefit from this investment.

* See next slide

What Are MCP Apps?

Interactive visual experiences that render directly inside AI chat — not separate web apps.

The Core Idea

MCP Apps let an MCP server return **interactive HTML UI** — dashboards, forms, visualizations — that renders inline in the conversation.

The user stays in their chat. No context switching. No browser tabs.

How It Works

💬 **User asks** a question in Claude, VS Code, or ChatGPT

↓

🔧 **MCP Server** processes the request and builds an interactive UI

↓

🖼 **MCP App renders** inline — the user interacts with it directly in the chat

↓

🔄 **Bidirectional** — the app can call MCP tools and the host pushes updates

What Could Fabric MCP Apps Look Like?

🔍 Troubleshooting Dashboard

Pipeline failure analysis with root cause, confidence scores, and one-click remediation — right in VS Code Copilot.

📊 Visual Pipeline Builder

Drag-and-drop data pipeline configuration rendered inline in Claude Desktop or ChatGPT.

📈 Capacity Monitor

Real-time capacity usage charts with alerting thresholds — embedded in any chat surface.

✅ Claude Desktop ✅ VS Code ✅ Goose ✅ Postman

📎 modelcontextprotocol.io/docs/extensions/apps
📎 github.com/modelcontextprotocol/ext-apps

How We Work: Parallel Streams

These aren't sequential phases — they're parallel streams. We already have agents today. As we strengthen the foundation, every layer gets more powerful.

STREAM 1

Strengthen the Foundation

Close API gaps. Fix throttling. Unify monitoring APIs. Ship CLI v2. Complete SPN coverage for all item types.

STREAM 2

Unlock Automation

Enable reliable CI/CD, IaC, and scripted workflows. Terraform 100% coverage. GitHub Actions integrations. Script item type.

STREAM 3

Light Up Agents

Ship Remote MCP Server. Interactive MCP Apps. Autonomous agent workflows. Customer MCP Tools. Agent identity & governance.

The compounding effect

Each stream **amplifies the others**. Better APIs make scripts more reliable. Better scripts make agents more capable. Better agents make Fabric accessible to more people. We invest in all three simultaneously.

Scenario 1: Developer Troubleshooting from VS Code

🥷 Persona: Data Engineer | 📍 Where: VS Code with GitHub Copilot | 🎯 Goal: Fix a failing pipeline

1. **Pipeline fails**
   Developer gets a notification in VS Code that the ETL-Finance-Daily pipeline has failed
2. **Check logs via CLI**
   Developer runs fab logs ETL-Finance-Daily.datapipeline --live to see a live feed of logs directly in the terminal
3. **Ask Copilot for help**
   Developer asks GitHub Copilot: "Help me understand why this pipeline failed"
4. **MCP App renders insights**
   Copilot uses Fabric's MCP tools to pull logs, analyze them, and renders an MCP App showing root cause: memory overflow with 87% confidence
5. **One-click fix**
   The MCP App shows remediation options. Developer clicks "Increase executor memory"
6. **Agent executes**
   With the developer's approval, the agent applies the fix via Fabric API and re-triggers the pipeline

✓ **Resolved**
Pipeline succeeds. Total time: minutes, not hours. Zero portal visits.

Layers used: Foundation (Logging API, CLI) → Programmatic (scriptable commands) → Agentic (MCP App, Copilot)

Yellow highlight = Work on our side

Scenario 2: Business User Creates a Report from Claude Desktop

👤 Persona: Business Analyst | 📍 Where: Claude Desktop (organization account) | 🎯 Goal: Quickly create and share a visual report

1. **Need arises**
   A business analyst needs to present quarterly sales data at a meeting in 2 hours
2. **Opens Claude Desktop**
   The organization has Claude Desktop with Fabric MCP connected. The analyst types: "I need a visual report of Q4 sales by region"
3. **MCP discovers workspace**
   Claude uses Fabric MCP tools to find the analyst's workspace and available datasets
4. **Builds the report**
   Claude creates a Power BI report with regional breakdowns, trend lines, and KPIs — all through the MCP interface
5. **Preview and refine**
   An MCP App renders a preview of the report inline. The analyst says "Add a comparison to Q3" — Claude updates it
6. **Embed and share**
   The analyst asks "Embed this in our team's SharePoint page" — Claude generates the embed code and deploys it

✓ **Ready to present**
A polished, interactive BI report — created without ever opening the Fabric portal or Power BI Desktop

Layers used: Foundation (APIs, workspace access) → Agentic (MCP tools, interactive experience)

Requires our MCP to be able to generate a new report + optionally a MCP app to render embedded reports.

Scenario 3: Autonomous CI/CD Agent Manages Deployments

🤖 Persona: Autonomous Deployment Agent | 📍 Where: GitHub Actions | 🎯 Goal: Deploy, validate, and monitor workspace changes

1. **Code merged**
   A developer merges a PR with updated data pipeline definitions into the main branch
2. **Agent triggered**
   GitHub Actions workflow fires — the autonomous Fabric deployment agent starts
3. **Deploy via CLI**
   Agent uses Fabric CLI to deploy updated items to the staging workspace: fab deploy --workspace staging --items pipelines/
4. **Run validation**
   Agent triggers test pipelines and monitors execution via Fabric monitoring APIs (or new CLI command)
5. **Promote to production**
   Tests pass. Agent promotes changes to production workspace using deployment APIs (or new CLI command)
6. **Continuous monitoring**
   Agent sets up monitoring hooks — if any deployed pipeline fails within 24 hours, it auto-rolls back and alerts the team

✓ **Full audit trail**
Every action logged under the agent's service principal (SPN) identity — fully traceable, governed, and compliant

Layers used: Foundation (APIs, CLI, Identity/SPN) → Programmatic (CI/CD pipeline, scripts) → Agentic (autonomous monitoring and rollback)

Scenario 4: Visual Pipeline Configuration via MCP App

👤 Persona: Analytics Engineer | 📍 Where: Claude Desktop | 🎯 Goal: Build a data pipeline without navigating the portal

1 **Describe the pipeline**
User tells Claude: "I need a pipeline that ingests CSV files from Blob Storage, transforms them with Spark, and loads into my Lakehouse"

2 **MCP discovers available resources**
Claude uses the Fabric MCP to list the user's workspaces, lakehouses, and available connections

3 **Visual pipeline builder renders as MCP App**
An MCP App — interactive drag-and-drop pipeline builder appears inline in Claude — showing source, transform, and destination stages

4 **User refines visually**
User drags a "data quality check" step between transform and load. Adjusts Spark pool settings. Adds a schedule trigger.

5 **Validate and preview**
Claude validates the pipeline config via the API and shows a sample data preview from the source

6 **Deploy with one click**
User clicks "Deploy" in the MCP App — Claude creates the pipeline in the target workspace via the Fabric API

✓ **Pipeline live**
Production-ready pipeline deployed. User configured it visually, without ever opening the Fabric portal.

Layers used: Foundation (APIs, workspace access) → Agentic (MCP App visual builder, Claude interaction)

Building Blocks: What We Need to Build

Key efforts across the three layers that enable the vision. This is a living list — teams can add their building blocks.

Foundation Layer

Internal API Tooling
Tools for Fabric teams to spec, mock, and document their APIs

CLI v2
Composable commands, scripting support, live log streaming

SDK Expansion
Python SDK GA, .NET SDK improvements

Identity & SPN
Service principal support for all item types

Monitoring APIs
Unified logging across all Fabric engines

Programmatic Layer

Terraform Coverage
100% coverage for GA items

CI/CD Toolkit
GitHub Actions, Azure DevOps integrations

Scripting in Fabric
Script item type, Copilot Script Mode

Agentic Layer

MCP Platform
Remote/Hosted MCP for external agents

MCP Apps
Interactive agent-rendered experiences

Fabric Unified MCP Server (TBD?)

Each building block maps to a specific scenario and layer. Teams can add their efforts to this framework.

Building Block: [Effort Name]

Owner	*[Team / PM Name]*
Layer	
Description	*[What is this effort and why does it matter?]*
Key Deliverables	*[List of concrete deliverables]*
Dependencies	*[What does this effort depend on?]*
Scenarios Enabled	*[Which end-to-end scenarios does this unlock?]*
Status	

Building Block: [Effort Name]

Owner	*[Team / PM Name]*
Layer	
Description	*[What is this effort and why does it matter?]*
Key Deliverables	*[List of concrete deliverables]*
Dependencies	*[What does this effort depend on?]*
Scenarios Enabled	*[Which end-to-end scenarios does this unlock?]*
Status	

Where We Think We Should Start

Four building blocks we propose to invest in first — each unlocks capabilities across the three-layer model.

FOUNDATION

API Gaps & Throttling

Close the top API gaps. Fix throttling for agent-scale traffic.

Prerequisite for everything else — scripts, agents, CI/CD all depend on complete, reliable APIs.

FOUNDATION

Internal API Tooling *

Build tools that help other Fabric teams complete their APIs faster.

As the platform team, we accelerate the entire Fabric ecosystem — not just our own surface.

AGENTIC

Troubleshooting MCP App

Proof-of-concept of the three-layer model, end-to-end.

Grounded in #1 CAT pain point (opp score 6.1). Demonstrates the full vision in one tangible experience.

PROGRAMMATIC

CLI v2 + Scripting

The composable execution backbone for both programmatic and agentic layers.

Open-source, extensible CLI framework. Powers scripts, CI/CD, and agent tool invocations.

⚠️ This Is Not a Full Plan Yet

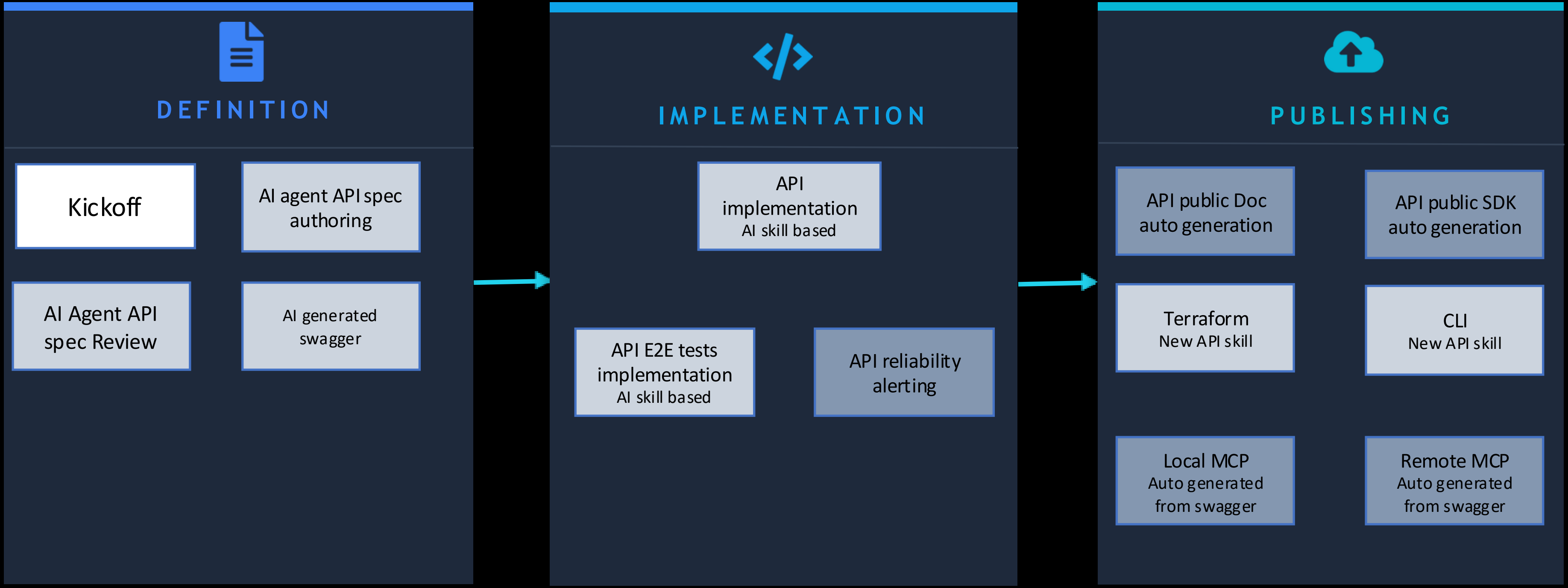
We first need alignment on the vision and the three-layer model. Once aligned, we scope, resource, and timeline these efforts together as a team.

✅ Why These Four?

They span all three layers, address the #1 customer pain point, and each one unlocks capabilities that compound across the entire strategy.

Public API Creation Process

Automation of public API creation – from PM spec to live public API


DEFINITION
Kickoff
AI agent API spec authoring
AI Agent API spec Review
AI generated swagger
IMPLEMENTATION
API implementation AI skill based
API E2E tests implementation AI skill based
API reliability alerting
PUBLISHING
API public Doc auto generation
API public SDK auto generation
Terraform New API skill
CLI New API skill
Local MCP Auto generated from swagger
Remote MCP Auto generated from swagger

AUTOMATION LEGEND

▨	Script-based automation
☐	AI-Assisted automation

What Real Customers Are Telling Us

Based on 56 hours of 1:1 interviews with professional Fabric developers (CAT Research, Nov 2025)

220
unique issues found

13
underserved jobs

6.1
top opportunity score (troubleshooting)

NPS 12
heavy users vs 42 overall

CI/CD and Deployment

Slow, manual deployments. Git integration gaps. No reliable pipeline automation.

Monitoring and Troubleshooting

Logs scattered across tools. Vague error messages. Hours to resolve issues.

Cost Optimization

No visibility into capacity costs. No automated cost controls.

Access and Identity

No service accounts. Artifacts break when users leave the org.

"Logs are scattered across multiple tools, making root cause analysis slow and error-prone." — **Enterprise Fabric customer**

Everyone Else Is **Already Moving**

AWS, Snowflake, and Databricks have all shipped MCP servers and are investing heavily in agent integration.

AWS

MCP servers for Bedrock, Lambda, S3

Agent-first workflows in production

Snowflake

MCP server (data-query only) shipping GA

Cortex AI layer over data estate

Databricks

MCP server for Unity Catalog

Developer-first design philosophy

82%
of orgs are API-first (Postman 2025)

70%
MCP-aware, only 10% using it — first-mover window

$52.6B
AI agent market by 2030

Competitors cover data-chat. We can own the full operational surface — if we move now.

Fabric is not just a portal. **It is the data layer embedded in every tool, workflow, and agent.**

Foundation	**Programmatic**	**Agentic**
APIs, CLI, SDKs, Identity	CI/CD, Scripts, IaC	MCP Apps, Autonomous Agents

The platforms that win are the ones that disappear into the workflow.

Fabric Embed and Automation Team • February 2026

Current efforts

Appendix

1. Slides of the brainstorming session

What CAT Research Told Us

Developer Usage (Dec 2025)

110K	API MAU (+13% MoM)
1.4K	CLI MAU (+17% MoM)
90%	of Fabric customers use automation
56 hrs	of 1:1 interviews with pro-devs

The Gap We Need to Close

Professional developers are Fabric's **most frequent** automation users...

...but also the **least satisfied**.

NPS 12 (heavy users) vs 42 overall

The Question

How do we enhance the lives of **professional developers** through intelligent automation?

We've identified five potential directions. Each stands alone, but they can be composed.

We want your input on prioritization, feasibility, and what we're missing.

Summary: The Five Directions

1	**Portal Scripting**	Automate click-ops without leaving Fabric
2	**VS Code Experiences**	Meet developers in their IDE with sandboxing
3	**Functions as MCP Tools**	Expose business logic to AI agents
4	**Externalizing Agentic**	External agents get internal Copilot power
5	**Blueprint Expansion**	IaC + policies + AI-assisted provisioning

Direction 1

Help Developers Within the Portal

Bringing developers INTO Fabric

The Idea

Introduce scripting capabilities directly in the Fabric portal so users can automate repetitive click-ops without leaving Fabric.

What This Could Look Like

- New **Script Item** type for storing CLI scripts
- **Copilot Script Mode**: describe task, get script
- Scripts are shareable, schedulable, version-controlled
- Runs with proper identity (SPNs)

Why It Matters

Unlocks the "long tail" of automation: tasks too niche for full features, too frequent to stay manual.

Example

Admin creates a weekly cleanup script that scans workspaces and emails owners of inactive ones — all from within Fabric.

Discussion Questions

- What are the top click-ops to target first?
- How do we balance simplicity vs. power?
- Security/governance implications?

Direction 2

Meet Developers Where They Are

Bringing Fabric TO developers (VS Code)

The Idea

Make Fabric a first-class experience in VS Code — local development, sandboxing, and seamless deployment.

What This Could Look Like

- **Fabric Sandbox**: Ephemeral workspaces with anonymized data
- **Local MCP**: Validate pipelines locally before deploying
- **Blueprint-to-Code**: Export configs as Terraform/CLI
- One-click deployment from IDE to cloud

Why It Matters

Context switching costs ~23 min/interruption. VS Code is the #1 IDE. Devs expect Git workflows and local iteration.

Example

Engineer spins up a sandbox with anonymized data, debugs pipeline locally using Local MCP, promotes to production only when ready.

Discussion Questions

- What's the right scope for "sandbox"?
- How do we handle data seeding?
- Integration with Git/CI/CD workflows?

Direction 3

Help Customers Expose Business Logic

Functions as MCP Tools

The Idea

Let customers publish their Fabric logic (notebooks, pipelines, stored procedures) as MCP tools that AI agents can discover and invoke.

What This Could Look Like

- Wrap a notebook as an MCP tool with name, description, params
- New **MCP Server Item**: container of custom tools
- Discoverable by Copilot and external agents
- Governed, auditable, secure

Aligns with Fabric IQ and the current AI/agent momentum

Example

Finance team wraps their proprietary "ComputeFinanceMetric" calculation as an MCP tool. Now Copilot (or any agent) can invoke it directly.

Discussion Questions

- What's the right abstraction? Item-level? Workflow?
- Discoverability: Marketplace? Registry?
- Security model for custom tools?

Direction 4

External Agents = Internal Power

Externalizing Fabric's Agentic Capabilities

The Idea

Fabric Copilot uses MCP tools internally. Let's expose those same capabilities to **external agents** — GitHub Copilot, Claude, Azure AI, custom agents.

What This Could Look Like

- **Hosted MCP Server**: Unified endpoint for all Fabric operations
- **Agent Identity**: SPNs, governance, audit trails
- MCP Apps
- Admin controls: approve/restrict access, scopes, read-only modes
- Same permission checks as human users

Key focus: Agent identity and governance

Examples

GitHub Copilot extension spins up Fabric resources as you code. An Azure AI agent publishes reports to Fabric as part of a larger workflow.

Discussion Questions

- How do we handle identity for external agents?
- What governance controls do admins need?
- How do we balance openness with security?

Direction 5

Expand Blueprints: Code, Policies, Beyond

AI-Powered Provisioning

The Idea

Expand Blueprints from "item templates" to full Infrastructure-as-Code with policies, governance, and external integrations.

What This Could Look Like

- **Blueprints as Code**: JSON/YAML/Terraform, version-controlled
- **Policy Integration**: Naming conventions, access controls, Azure policies
- **AI-Generated Blueprints**: Natural language to workspace config
- Connect to things **outside Fabric**: Azure policies, Entra ID, compliance

20+ manual steps to 1 command

Example

"Create a workspace for the finance team with standard policies" → AI generates Blueprint with naming, roles, Azure policy compliance, pre-configured items. One command deploy.

Discussion Questions

- How do we integrate with Azure policies / external governance?
- What's the right level of AI assistance?
- Cross-workspace / cross-environment scenarios?